Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES MATHÉMATIQUES

Une étude de la nouvelle méthode des asymptotes mobiles en optimisation

DE BAETS, Vinciane

Award date:
1987

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Facultés Universitaires Notre Dame de la Paix
Namur
Faculté des Sciences

Une étude de la nouvelle méthode des asymptotes mobiles en optimisation

Mémoire présenté pour l'obtention du grade de Licencié en Sciences Mathématiques par :

Vinciane DE BAETS

Promoteur : NGUYEN Van Hien
Directeur : DUBOIS Roger, Direction Technologies, Société SOLVAY & Cie.

Année 1986 - 1987

Facultès Universitaires Notre-Dame de la Paix
Faculté des Sciences
rue de Bruxelles 61,B-5000 **NAMUR**
Tel. 081-22.90.61 Telex 59222 facnam-b Telefax 081-23.03.91

Une etude de la nouvelle methode des asymptotes mobiles en optimisation.

DE BAETS Vinciane

Rèsumé
Dans ce travail, nous avons traitè de la mèthode des asymptotes mobiles qui fut proposèe par K.SVANBERG du "Royal Institute of Technology", Stockholm, pour rèsoudre des problèmes non linèaires en structure mècanique par le biais d'une suite de sous problèmes approximants convexes sèparables.
Nous prèsentons ici une ètude de la convergence mathèmatique de la mèthode. Nous donnons ègalement une implèmentation de cette mèthode, qui s'inscrit dans une perspective d'utilisation industrielle.

Abstract
In this paper, we are concerned with the moving asymptots method that has been proposed by K.SVANBERG of "Royal Institute of Technology", Stockholm, for solving nonlinear problems arising in structural mechanics by successively approximating it by sequence of convex separable problems.
We give a mathematical convergence analysis of the method. We also present an implementation of the method in a industrial environment.

Mèmoire de licence en Sciences Mathèmatiques
Juin 1987
Unitè d'Optimisation
Promoteur : **Prof.NGUYEN.V.H.**
Directeur : **Docteur R.DUBOIS**, Direction Technologies ,
Societè SOLVAY Cie.

Je voudrais remercier personnellement, pour m'avoir guidée et aidée, Monsieur V. H. Nguyen, de l'unité d'Optimisation, FNDP, pour la partie mathématique et Monsieur E. Vergison et Monsieur R. Dubois, de la Direction Technologies, Société SOLVAY & Cie, pour la partie informatique.

Je remercie également les membres de l'unité d'Optimisation, et plus particulièrement Monsieur J. J. Strodiot pour sa disponibilité et ses conseils, et enfin, Monsieur F. Valette, de l'Atelier de Dessin, pour s'être occupé du graphisme.

Le Département de Mathématique des Facultés et la Société SOLVAY ont mis à ma disposition le matériel informatique nécessaire à la réalisation de ce mémoire, je les en remercie.

Table des matières.

Conclusion.

Annexe : Programme.

guide de l'utilisateur.

1e Partie : - Routine d'optimisation.
- Routines et fonctions utilisées.

2e partie : - Routines de contrôle.
- Interpréteur.

Références.

Chapitre 1 :

Introduction .

§ 1.1 : Type de problèmes traités :

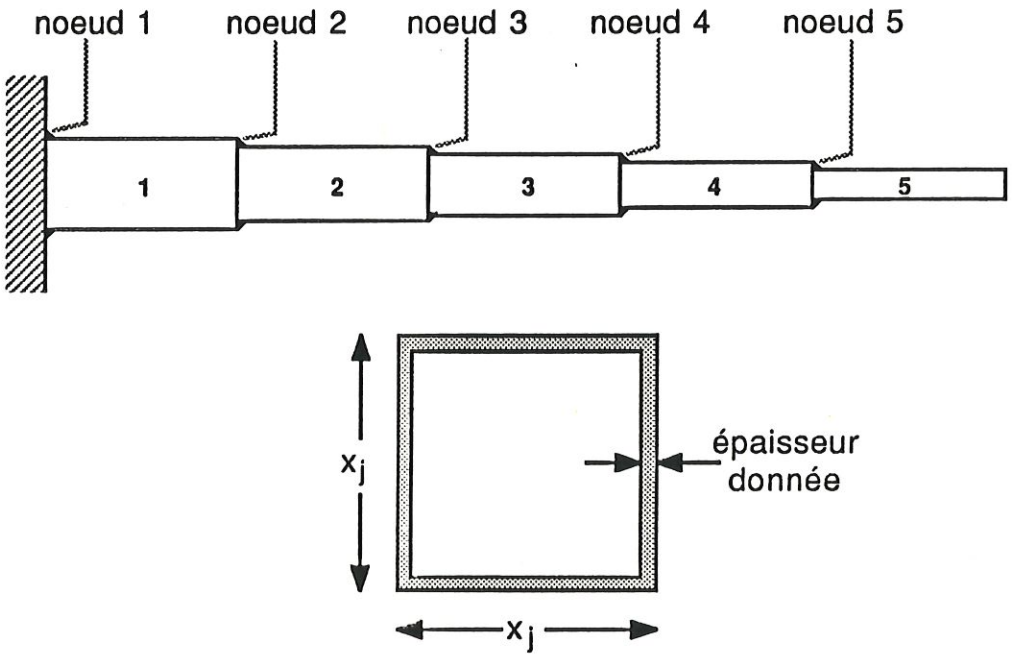
La méthode que nous allons étudier concerne l'optimisation de structure. C'est pourquoi, elle doit être relativement flexible et générale. En effet, les problèmes traités dans ce cadre font intervenir non seulement des variables représentant des tailles de composants du système mais aussi d'autres à propos de la forme de ces composants, ou encore à propos d'orientation d'angles entre ceux-ci. De même, une telle méthode doit être capable de traiter des contraintes de tous types pourvu que nous disposions des dérivées partielles de ces dernières que ce soit analytiquement ou numériquement.

Par conséquent, cette méthode s'insère dans la programmation non linéaire générale.

Mais il est intéressant aussi d'essayer de tenir compte de certaines caractéristiques qu'ont les problèmes en optimisation de structure. Les évaluations de fonctions reviennent, en général, très chères. Mais, d'autre part, il nous est possible de calculer les gradients des fonctions qui sont relativement peu coûteux.

A titre d'exemples, voici différents problèmes "classiques" en optimisation de structure.

1. Poutre cantilever.



La barre est rigidement fixée au noeud 1. Une force verticale externe fixée est appliquée au noeud 6 (extrémité de la barre).

Les variables représentent les poids des différents éléments dont l'épaisseur est donnée.

La fonction à minimiser est le poids du système.

Dans ce cas précis, il n'y aura qu'une contrainte indiquant une limite au déplacement vertical autorisé au noeud 6, dû à la force qui y est appliquée.

Les bornes sur les variables ne deviendront jamais actives, nous pouvons donc ne pas y prêter grande attention.

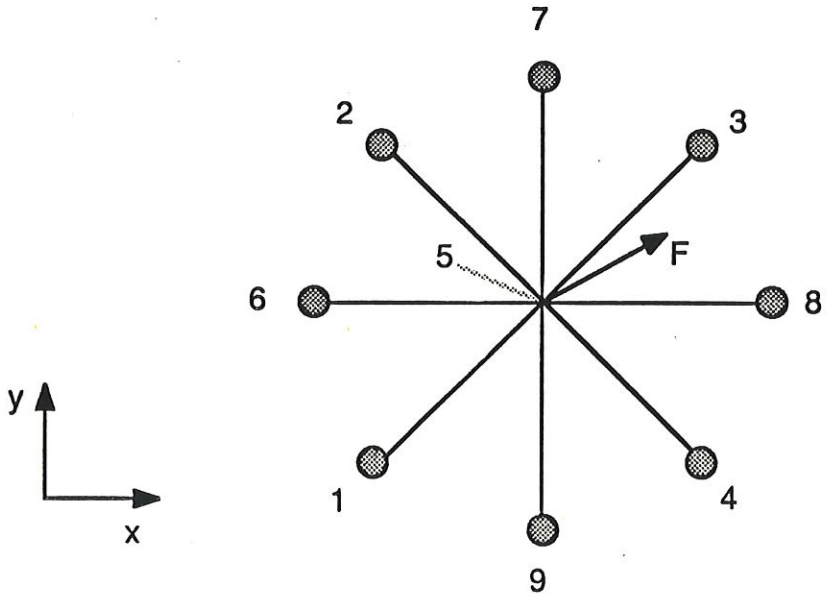
Ceci peut se modéliser de la façon suivante :

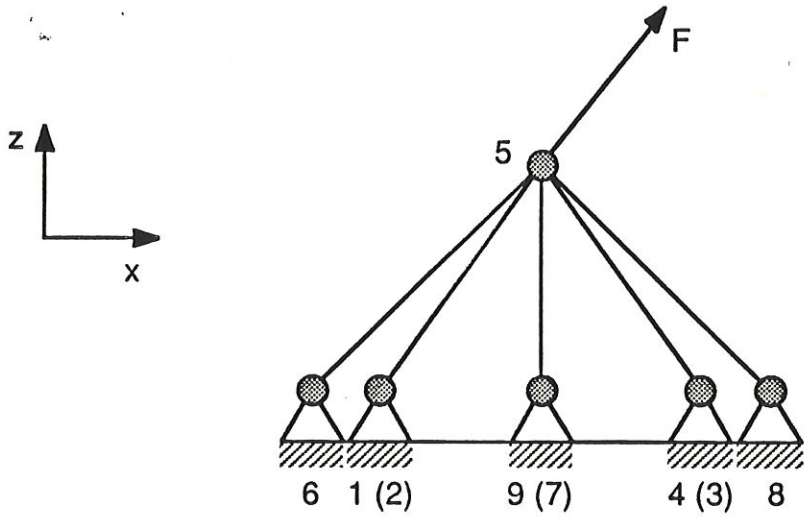
$$\begin{cases} \text{minimiser } c_1 (x_1 + x_2 + x_3 + x_4 + x_5) \\ \text{s.c. } \dfrac{61}{x_1^3} + \dfrac{37}{x_2^3} + \dfrac{19}{x_3^3} + \dfrac{7}{x_4^3} + \dfrac{1}{x_5^3} \leq c_2 \\ x_i > 0 \qquad i = 1, \ldots, 5 \end{cases}$$

où les constantes c_1 et c_2 dépendent des propriétés du matériau utilisé ainsi que de l'amplitude de la force appliquée.

2. Treillis à 8 barres.

Considérons la structure suivante composée de huit éléments :





Une force externe $F = (F_x, F_y, F_z)$ est appliquée au noeud 5, telle que, par exemple :

$F_x = 40$ kN $\quad F_y = 20$ kN $\quad F_z = 200$ kN

Les variables correspondent aux surfaces des sections transversales des éléments.

Les bornes inférieures sont fixées à 100 mm² tandis que les bornes supérieures sont si grandes qu'elles ne deviennent jamais actives.

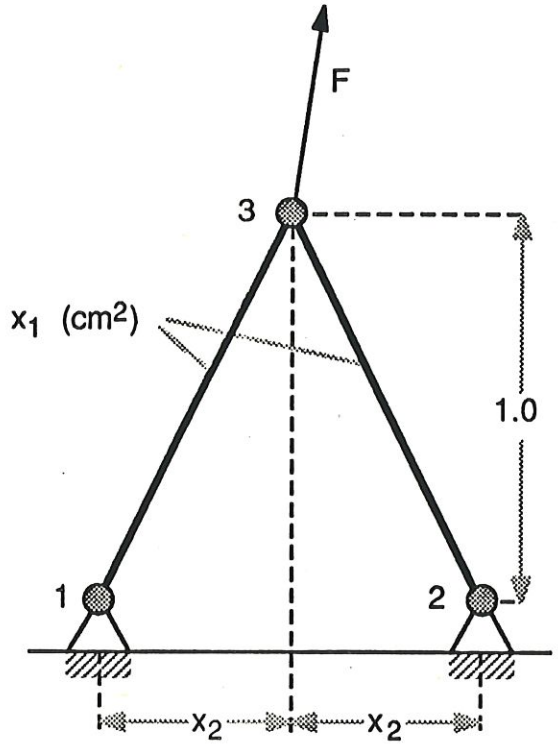
La fonction que nous cherchons à minimiser est le poids de la structure.

Les seules contraintes imposées sont des limites aux forces (de pression ou de tension) sur chacun des éléments. Ces forces sont déterminées par la force extérieure considérée.

3. Treillis à 8 barres.

Voici une structure composée de deux éléments. Nous y appliquons une force externe (F_x, F_y) au noeud 3, où $F_x = 24.8$ kN

$F_y = 198.4$ kN



Ce problème peut, en un sens, être couplé avec le précédent mais il présente cependant un intérêt différent. En effet, nous remarquons ici des variables représentant des surfaces de sections et d'autres, des distances, contrairement au cas du treillis à 8 barres.

Ainsi, la variable x_2 correspond à la

surface de la section transversale des éléments et la variable x_2, à la moitié de la distance séparant les noeuds 1 et 2.

La fonction à minimiser est le poids du système.

Les contraintes imposent que la force de tension dont sont l'objet les deux éléments, ne peut dépasser 100 N/mm².

Ceci peut s'exprimer analytiquement comme suit :

$$\begin{cases} \text{minimiser } w(x_1, x_2) = c_1 x_1 \sqrt{1 + x_2^2} \\ \text{s.c. } \sigma_1(x_1, x_2) = c_2 \sqrt{1 + x_2^2} \left(\dfrac{8}{x_1} + \dfrac{1}{x_1 x_2} \right) \leq 1 \\ \sigma_2(x_1, x_2) = c_2 \sqrt{1 + x_2^2} \left(\dfrac{8}{x_1} - \dfrac{1}{x_1 x_2} \right) \leq 1 \\ 0.2 \leq x_1 \leq 4 \\ 0.1 \leq x_2 \leq 1.6 \end{cases}$$

où $c_1 = 1$, $c_2 = 0.124$

A la vue de ces exemples types, nous pouvons déduire la forme générale du problème que nous voulons traiter.

$$\begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f_i} \quad i = 1, \dots, m \\ & \underline{x}_j \leq x_j \leq \overline{x}_j \quad j = 1, \dots, n \end{cases}$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$

(Au niveau des notations, les indices inférieurs sur les variables dénotent les composantes en question.)

Il est à noter qu'aucune hypothèse ou contrainte n'est imposée sur les paramètres $\hat{f_i}$. Dans la même idée, les bornes peuvent prendre n'importe quelle valeur et même éventuellement, une valeur nulle ou négative.

Ceci nous permet de garder un caractère assez général et ne nous restreint pas à l'éventail des problèmes que nous sommes capables de résoudre.

§ 1.2 : Méthode itérative de résolution.

La méthode proposée consiste en un processus itératif comme c'est généralement le cas en programmation non linéaire.

Nous allons donc construire et résoudre une suite de sous-problèmes jusqu'à ce qu'un critère de convergence soit vérifié, ou plus simplement, jusqu'à ce que le résultat intermédiaire obtenu nous donne satisfaction.

Nous pouvons modéliser cela par l'algorithme suivant :

Pas 0 : Initialisation : - choix du point de départ x^0.
- poser $k = 0$.

Pas 1 : Soit x^k.
Construction du sous-problème $\overline{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\overline{P}(x^k)$.
Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$.
Retourner au pas 1.

(Pour les notations, l'indice supérieur sur les

variables dénote l'itération à laquelle nous nous trouvons.)

Remarquons que cette structure est commune à de nombreuses méthodes. Ce qui les différencie, c'est la manière de construire les sous-problèmes $\overline{P}(x^k)$.

Ainsi, par exemple, L.A. Schmit et B. Farshi [1] ont proposé une méthode de ce type. Les sous-problèmes $\overline{P}(x^k)$ étaient obtenus en linéarisant les fonctions des contraintes par rapport aux variables réciproques, tandis que la fonction objectif restait inchangée.

Plus récemment, cette façon de procéder a été généralisée par C. Fleury et V. Braibant [2] pour pouvoir élargir le domaine d'application et pouvoir ainsi traiter des problèmes où interviennent des variables autres que celles à propos de la taille des éléments en jeu. Cette fois-ci, la linéarisation par rapport aux variables réciproques n'est plus automatiquement appliquée. La méthode tient compte du signe des dérivées partielles des fonctions et suivant celui-ci, les linéarisations se font par rapport

aux variables ou aux variables réciproques. Nous parlons alors de linéarisation mixte. Pour une analyse mathématique de cet algorithme, il est intéressant de consulter, par exemple, un article de V.H. Nguyen, J.J. Strodiot et C. Fleury [3].

La méthode des asymptotes mobiles s'insère dans cette perspective et peut être interprétée comme une généralisation de la précédente. Mais, étant plus flexible, elle sera plus performante. En effet, elle fait intervenir des paramètres supplémentaires appelés "asymptotes mobiles" qui permettent de stabiliser et augmenter la vitesse de convergence du processus général. L'intérêt consiste aussi dans le fait qu'aucune contrainte de non-négativité n'est imposée sur les variables et cela, contrairement à la méthode de C. Fleury et V. Braibant.

§ 1.3 : Asymptotes mobiles.

Nous allons ici développer le principe sur lequel la méthode est basée. L'idée de Svanberg [4] est assez simple. Elle consiste

à associer à un point donné deux paramètres.

Ainsi, à un point $x \in \mathbb{R}^n$, nous ferons correspondre $U \in \mathbb{R}^n$ et $L \in \mathbb{R}^n$ où L et U seront appelés "asymptotes mobiles" et seront tels que $L < x < U$.

Donc, à tout point de l'espace $\mathbb{R}^n$, sera associé une paire $\{L, U\}$. Il nous reste alors, en un premier temps, à préciser la manière dont seront construites les asymptotes et ensuite, ce qui est l'intérêt de la méthode, leur utilité.

En ce qui concerne le choix des asymptotes, plusieurs possibilités peuvent être envisagées.

Si nous nous rappelons le problème P que nous avons à résoudre :

$$\begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots n \end{cases}$$

nous remarquons que nous allons plus particulièrement nous intéresser à un intervalle précis $[\underline{x}, \bar{x}]$.

Nous pouvons noter que ces bornes sur les variables vont intervenir lors de la construction

de L et U.

Un premier choix consiste à procéder comme suit:

$$\begin{cases} L_j = \underline{x}_j - s_0 (\bar{x}_j - \underline{x}_j) \\ U_j = \bar{x}_j + s_0 (\bar{x}_j - \underline{x}_j) \end{cases} \qquad j = 1 \dots n$$

où s_0 est un nombre réel fixé (par exemple $s_0 = 0.1$)

Analysons le procédé. Etant donné que la méthode est itérative, nous allons obtenir une suite d'itérés $(x^k)_{k \in \mathbb{N}}$. Vu la façon dont nous avons défini les asymptotes, quel que soit l'itéré considéré x^k, il lui sera associé la même paire $\{L, U\}$. Ceci provient du fait que le point considéré n'apparaît pas explicitement dans l'expression de L et U. Dans ce sens, nous pouvons dire que les asymptotes obtenues ainsi seront peu tôt fixes et assez peu mobiles.

Nous pouvons désirer nous débarrasser de cette rigidité et introduire alors une dépendance directe envers le point considéré.

Ainsi, à l'itéré x^k, nous associons:

$$\begin{cases} L_j^k = x_j^k - (\bar{x}_j - \underline{x}_j) \\ U_j^k = x_j^k + (\bar{x}_j - \underline{x}_j \end{cases} \qquad j = 1 \dots n$$

Il est clair que, dans ce cas, les paires $\{L^k, U^k\}$ correspondant aux divers itérés x^k ne seront plus identiques. C'est pourquoi, nous pouvons dire que ces asymptotes sont "mobiles".

Cependant, si nous voulons augmenter au maximum l'efficacité de la méthode, il est intéressant lors de la construction de $\{L^k, U^k\}$ associés à x^k, de tenir compte de la façon dont le processus itératif s'est comporté dans les itérations antérieures. Et plus particulièrement, considérons l'évolution des itérés x^{k-2} et x^{k-1}.

Le raisonnement consiste à étudier la monotonie ou la tendance qu'a la méthode à osciller.

Si elle oscille, nous devons alors essayer de la stabiliser. Comme nous le développerons plus tard, ceci peut s'effectuer en rapprochant les asymptotes.

Tandis que si la méthode est monotone et lente, elle peut être relaxée. Nous écartons, dans ce cas, les asymptotes.

Ceci peut s'exprimer comme suit:

Pour $k = 0$ et $k = 1$:

$$\begin{cases} L_j^k = x_j^k - (\bar{x}_j - \underline{x}_j) \\ U_j^k = x_j^k + (\bar{x}_j - \underline{x}_j) \end{cases} \qquad j = 1 \dots n$$

Pour $k \geq 2$:

- s'il y a une oscillation i.e. :

$$(x_j^k - x_j^{k-1}) \cdot (x_j^{k-1} - x_j^{k-2}) < 0$$

$$\begin{cases} L_j^k = x_j^k - s\,(x_j^{k-1} - L_j^{k-1}) \\ U_j^k = x_j^k + s\,(U_j^{k-1} - x_j^{k-1}) \end{cases} \qquad j = 1 \dots n$$

- s'il y a convergence monotone i.e. :

$$(x_j^k - x_j^{k-1}) \cdot (x_j^{k-1} - x_j^{k-2}) \geq 0$$

$$\begin{cases} L_j^k = x_j^k - \dfrac{(x_j^{k-1} - L_j^{k-1})}{s} \\ U_j^k = x_j^k + \dfrac{(U_j^{k-1} - x_j^{k-1})}{s} \end{cases} \qquad j = 1 \dots n$$

où s est un nombre réel donné < 1 .

(Par exemple : $s = 0.7$)

Ce ne sont, bien sûr, que des idées ou des suggestions de construction et il est possible de procéder différemment du moment que soit vérifié :

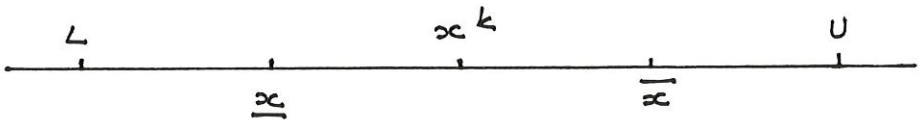
$$L_j^k < x_j^k < U_j^k \qquad j = 1 \dots n$$

Attachons-nous maintenant à dégager l'utilité de ces asymptotes. Nous avons vu, précédemment, que pour le problème P, nous devions nous restreindre à l'intervalle $[\underline{x}, \overline{x}]$. Les asymptotes vont nous permettre de jouer sur cet intervalle et de n'en considérer qu'une partie. C'est pourquoi, il sera nécessaire d'exiger $L_j < \overline{x}_j$ et $\underline{x}_j < U_j$, $j \in \{1 \dots n\}$, pour que les intervalles $[\underline{x}, \overline{x}]$ et $[L, U]$ aient une intersection non vide. Ceci nous évite de "vider" le domaine admisible.

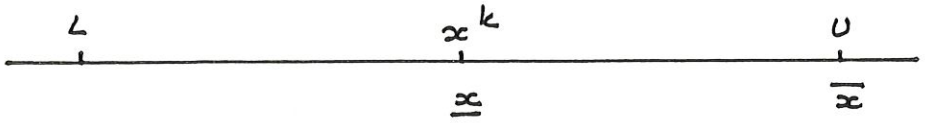
Représentons les diverses situations qui peuvent alors se produire à une dimension.

Pour le premier choix de construction, nous obtiendrions :

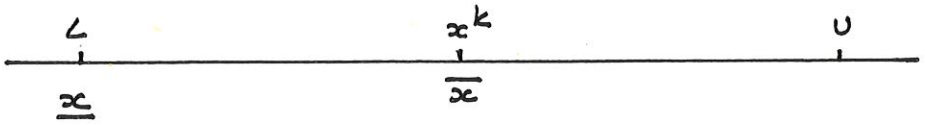
Pour le deuxième choix de construction, nous obtiendrions :



ou bien :



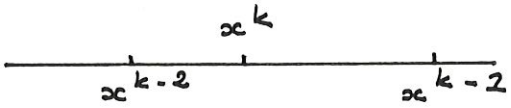
ou encore :



Nous voyons ainsi certaines positions relatives que peuvent occuper les paramètres L et U vis-à-vis de $\underline{x}$ et $\overline{x}$.

Il est surtout intéressant d'analyser et d'interpréter la troisième façon de construire L et U.

Quand le processus oscille, nous le stabilisons en rapprochant les asymptotes. En effet, s'il se produit une oscillation :



nous pouvons alors situer le point vers lequel la méthode se dirige, à "l'intérieur" de l'oscillation. Ainsi, en rapprochant les asymptotes, nous réduisons l'intervalle $[L, U]$ et donc aussi son intersection avec $[\underline{x}, \overline{x}]$.

nous diminuons l'espace où nous allons travailler pour mieux cerner le point vers où nous semblons nous diriger.

Par contre, si le processus évolue lentement de façon monotone, il est très probable que ce sont les asymptotes qui le ralentissent et donc, en les écartant, nous augmentons l'intervalle des points "admissibles" (pour les contraintes de bornes). cela permet d'accélérer le processus.

Cette dernière analyse nous montre la flexibilité de la méthode et nous permet de voir en quoi le troisième choix de construction de L et de U présente peu d'intérêt que les deux autres.

§ I.4 : Plan du mémoire.

Dans les chapitres qui suivent, nous avons développé la méthode des asymptotes mobiles présentée récemment par le docteur K. Svanberg du "Royal Institue of Technology" à Stockholm [4].

Plus précisément, nous avons d'abord dans le chapitre 2, détaillé la manière dont nous

pouvons construire des approximations de fonctions à partir des asymptotes mobiles. Les propriétés ont été démontrées en annexe.

Ensuite, dans le chapitre 3, nous avons étudié les sous-problèmes engendrés à chaque itération et les ajustements qui se sont avérés nécessaires pour garder la cohérence de la méthode.

Dans le chapitre 4, nous nous sommes attachés à présenter une démonstration de la convergence de l'algorithme. Ce point n'ayant pas du tout été soulevé par K. Svanberg dans son article [4], il nous a paru, cependant, intéressant de s'y attarder. Nous avons donc entrepris, personnellement, une étude à ce sujet : ce qui constitue essentiellement l'originalité du présent travail.

Ensuite, d'un point de vue plus pratique, nous avons envisagé, dans le chapitre 5, les problèmes causés par un mauvais choix du point de départ et quels remèdes nous pouvions y apporter.

Pour exploiter la structure particulière des sous-problèmes, nous les résolvons par dualité. C'est pourquoi, dans le chapitre 6, nous avons

analysé le duale et sommes arrivés à des expressions explicites des variables primales, gradient et Hessien de la fonction duale.

Quant à l'implémentation, nous avons, tout d'abord, programmé une routine réalisant l'optimisation suivant la méthode étudiée. Et ensuite, nous nous sommes consacrés à son "habillement" informatique (interpréteur, commandes, ...) pour nous intégrer dans une perspective industrielle. Cette partie effectuée avec la collaboration de la société Solvay, constitue également l'originalité du travail. Ceci est détaillé dans le chapitre 7.

Pour terminer, nous avons donné quelques résultats numériques que nous avons obtenus lors de tests sur le Cluster Vax des Facultés.

Chapitre 2 :

"Linéarisation".

§ 2.1 : "Linéarisée par rapport aux variables réciproques" d'une fonction.

Considérons une fonction $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et un point $x \in \mathbb{R}^n$.

Nous allons, dans ce chapitre, essayer de construire une approximation de f basée sur le point x.

Dans le chapitre précédent, nous avons vu comment associer à tout point de $\mathbb{R}^n$, une paire $\{L, U\}$ où les paramètres L et U sont appelés "asymptotes mobiles".

Construisons alors la paire $\{L, U\}$ correspondant au point x.

Définissons ensuite, l'approximation comme suit : nous "linéarisons" f autour de x par rapport aux variables $\dfrac{1}{y_j - L_j}$ et $\dfrac{1}{U_j - y_j}$

(où l'indice j indique la composante). [4]

Remarquons que le processus n'est pas automatique. En effet, nous tenons compte des signes des dérivées partielles de f.
Ainsi, si $\left.\dfrac{\partial f}{\partial y_j}\right|_x < 0$, nous considérons la variable $\dfrac{1}{y_j - L_j}$.

Par contre, si $\frac{\partial f}{\partial y_j}\Big|_x \geq 0$, nous considérons la variable $\frac{1}{v_j - y_j}$.

La particularité de cette approximation consiste à ne linéariser que par rapport à des variables réciproques.

Nous noterons, par la suite, $\overline{f}(y;x)$, la linéarisée par rapport à $\frac{1}{v_j - y_j}$ et $\frac{1}{y_j - z_j}$ de la fonction f en le point $x \in \mathbb{R}^n$.

Quelle est alors l'expression de $\overline{f}(y;x)$?

Après changement de variables et linéarisation (cfr. Annexe 2. A.), nous obtenons :

$$\overline{f}(y;x) = f(x) + \sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x + \sum_{j \in J-} \left[(x_j - z_j) - \frac{(x_j - z_j)^2}{y_j - z_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

où $\begin{cases} J+ = \{ j \in \{1, \dots, n\} \text{ tels que } \frac{\partial f}{\partial y_j}\Big|_x \geq 0 \} \\ J- = \{ j \in \{1, \dots, n\} \text{ tels que } \frac{\partial f}{\partial y_j}\Big|_x < 0 \} \end{cases}$

§ 2.2 : Convexité.

Une propriété remarquable de l'approximation $\overline{f}(y;x)$ d'une fonction f autour d'un point x, est la convexité.
(cfr. Annexe 2.B.)

§ 2.3 : Approximation exacte.

Nous notons aussi que cette approximation $\overline{f}(y;x)$ est exacte au point autour duquel elle s'effectue, i.e. le point x. Et nous pouvons dès lors écrire :

$$\overline{f}(x;x) = f(x)$$

Cela peut se vérifier très facilement. Il suffit de considérer l'expression de $\overline{f}(y;x)$:

$$\overline{f}(y;x) = f(x) +$$

$$\sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J-} \left[(x_j - l_j) - \frac{(x_j - l_j)^2}{y_j - l_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

où $J+$ et $J-$ ont été définis plus haut.

Soit $y \equiv x$

$$\overline{f}(x;x) = f(x) +$$

$$\sum_{j \in J+} \left[\frac{(u_j - x_j)^2}{u_j - x_j} - (u_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J-} \left[(x_j - l_j) - \frac{(x_j - l_j)^2}{x_j - l_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$= f(x)$$

Cela nous donne, de suite, le résultat.

Cette approximation est donc convexe et exacte en x. Il est intéressant de voir que nous avons aussi cette dernière propriété en ce qui concerne les gradients

i.e. $\nabla \overline{f}(x;x) = \nabla f(x)$

Cela se vérifie tout aussi aisément à partir de l'expression de $\nabla \overline{f}(y;x)$:

$$\left(\nabla \overline{f}(y;x)\right)_j = \begin{cases} \dfrac{(u_j - x_j)^2}{(u_j - y_j)^2} \dfrac{\partial f}{\partial y_j}\Big|_x & \text{si } j \in J+ \\ \dfrac{(x_j - l_j)^2}{(y_j - l_j)^2} \dfrac{\partial f}{\partial y_j}\Big|_x & \text{si } j \in J- \end{cases}$$

En $y \equiv x$, nous obtenons :

$$\left(\nabla \overline{f}(x ; x) \right)_j = \frac{\partial f}{\partial y_j}\Big|_x \qquad \forall j$$

Et donc $\nabla \overline{f}(x ; x) = \nabla f(x)$.

§ 2.4 : Cas particuliers.

Cette façon d'approximer une fonction f autour d'un point x possède un intérêt particulier en ce sens qu'elle peut être considérée comme une généralisation de linéarisations plus classiques telles que la linéarisation directe ou la linéarisation convexe. En effet, ceci découle du principe même sur lequel se base la construction de l'approximation.

Rappelons-nous qu'à tout point de $\mathbb{R}^n$, nous pouvons associer une paire $\{L, U\}$ d'asymptotes mobiles. Dans le chapitre précédent, nous avons développé diverses méthodes pour définir ces paramètres et nous avons signalé qu'elles n'étaient pas exhaustives et que nous pouvions envisager d'autres procédures.

Nous avons vu, ensuite, comment à partir de ces asymptotes, nous établissons l'approximation. Remarquons donc la dépendance "directe" de l'approximation vis-à-vis des asymptotes choisies.

En effet, soit une fonction f quelconque et un point $x \in \mathbb{R}^n$. Construisons $\{ \overline{L}, \overline{U} \}$ associée à x et $\overline{f}(y ; x)$. Considérons, ensuite, une autre paire $\{ \overline{\overline{L}}, \overline{\overline{U}} \}$ toujours associée à x mais obtenue par un procédé différent. Nous aurons une autre approximation $\overline{\overline{f}}(y ; x)$ qui ne sera pas identique à la précédente.

Voyons par exemple :

soit $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ $\quad f(y) = 5y_2 - y_1^2 - 10$

soit $x = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

soit $\{ \overline{L} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\overline{U} = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \}$

alors $\overline{f}(y ; x) = -48 + \dfrac{4}{y_1 - 2} + \dfrac{320}{10 - y_2}$

soit $\{ \overline{\overline{L}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\overline{\overline{U}} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \}$

alors $\overline{\overline{f}}(y ; x) = -23 + \dfrac{4}{y_1 - 2} + \dfrac{45}{5 - y_2}$

Ceci nous permet d'affirmer que le choix des asymptotes correspondant au point x (appelé point de linéarisation), influence très fortement l'approximation calculée.

Nous pouvons donc posséder plusieurs approximations différentes d'une même fonction f autour d'un même point x.

Il est possible d'exploiter cette propriété. Ainsi, pour des valeurs particulières de L et U, nous retrouvons des linéarisations "classiques".

2.4.1 : Linéarisation directe.

Suivant l'idée exposée ci-dessus, il nous est possible, à partir de $\overline{f}(y;x)$ pour des valeurs bien précises de L et U, de retrouver la linéarisation directe.

Il suffit pour cela de considérer :

$$L \longrightarrow -\infty$$

$$U \longrightarrow +\infty$$

Ce résultat peut sembler, à priori, assez surprenant, étant donné que, comme nous l'avons développé plus haut, pour obtenir l'expression de $\overline{f}(y;x)$, nous effectuons une

linéarisation par rapport à $\frac{1}{U_j - y_j}$ ou $\frac{1}{y_j - L_j}$

qui font intervenir des variables réciproques. Mais ce résultat peut, cependant, être montré assez aisément en utilisant les règles de l'Hospital (cfr. Annexe 2.C.)

2.4.2 : Linéarisation convexe.

Suivant le même principe, il est possible par un choix judicieux des asymptotes L et U, de se ramener à la linéarisation convexe.

Il convient pour cela d'avoir :

$$L = 0$$

$$U \longrightarrow +\infty$$

(cfr. Annexe 2.D.)

§ 2.5 : Conservativité.

Nous avons établi que, pour une fonction f, il nous est possible de construire plusieurs approximations autour d'un même point, suivant le choix des asymptotes.

Il est intéressant alors de pouvoir comparer ces approximations pour pouvoir voir si les unes ont plus d'intérêt que les autres.

Par exemple, nous pourrions nous poser la question de la conservativité.

A ce propos, nous pouvons établir une propriété qui nous permet de voir comment sélectionner une approximation pour qu'elle soit plus conservative qu'une autre.

En effet, nous pouvons démontrer que :
(cfr. Annexe 2.E.)

Soient $\overline{f}(y;x)$ et $\overline{\overline{f}}(y;x)$, 2 approximations de f autour de x construites sur les asymptotes $\{\overline{L}, \overline{U}\}$ et $\{\overline{\overline{L}}, \overline{\overline{U}}\}$

si $\quad \overline{L}_j \leq \overline{\overline{L}}_j < x_j < \overline{\overline{U}}_j \leq \overline{U}_j \qquad j = 1 \dots n$

i.e. si les asymptotes sont emboîtées les unes dans les autres

alors $\quad \forall\, y \in \mathbb{R}^n$ tel que

$$\overline{\overline{L}}_j < y_j < \overline{\overline{U}}_j \qquad j = 1 \dots n$$

nous aurons

$$\overline{\mathcal{F}}(y ; x) \leq \overline{\overline{\mathcal{F}}}(y ; x)$$

Et donc, $\overline{\overline{\mathcal{F}}}(y ; x)$ sera plus conservative que $\overline{\mathcal{F}}(y ; x)$ sur l'intervalle $]\overline{\overline{L}}, \overline{\overline{U}}[$.

En effet, la condition

$$\overline{\overline{\mathcal{F}}}(y ; x) \leq \hat{\mathcal{F}}$$

entraîne alors

$$\overline{\mathcal{F}}(y ; x) \leq \hat{\mathcal{F}}$$

et le premier domaine d'admissibilité est inclus dans le second.

Grâce à ce résultat, nous pouvons aisément arriver à la conclusion selon laquelle la linéarisation convexe est plus conservative que la linéarisation directe comme cela a déjà été établi [2].

En effet, la linéarisation directe correspond au choix de $\overline{L} \to_0 -\infty$ et $\overline{U} \to_0 +\infty$ et la linéarisation convexe au choix de $\overline{\overline{L}} = 0$ et $\overline{\overline{U}} \to_0 +\infty$.

Nous avons bien $\overline{L} \leq \overline{\overline{L}}$ et $\overline{\overline{U}} \leq \overline{U}$

Donc, sur $]0, +\infty[$, la linéarisation convexe sera plus conservative.

Nous pouvons aussi en déduire qu'il est

ponible de trouver d'autres approximations encore plus conservatives. Il suffit, pour cela, de considérer $\underline{\underline{L}}$ et $\overline{\overline{U}}$ tels que

$$0 \leq \underline{\underline{L}} \qquad \text{et} \qquad \overline{\overline{U}} \leq +\infty$$

Alors, sur l'intervalle $]\underline{\underline{L}}, \overline{\overline{U}}[$, la nouvelle approximation sera la plus conservative.

Si nous reprenons l'exemple étudié précédemment, ceci peut se visualiser sur un graphique :

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R} \qquad f(y) = 5y_2 - y_1^2 - 10$$

Soit $x = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

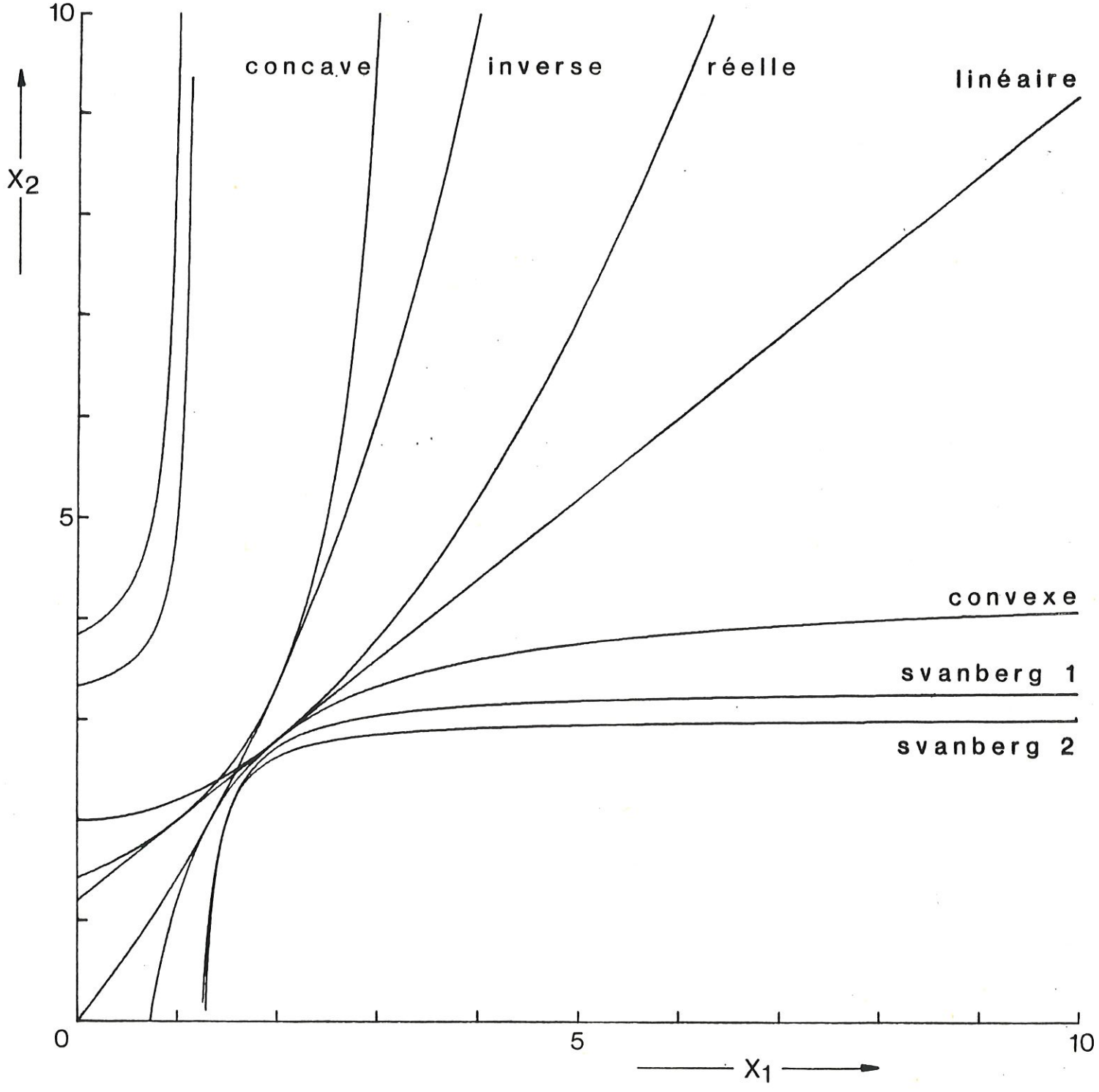
Nous avions obtenu :

(Svanberg 1) $\overline{f}(y; x) = -48 + \dfrac{4}{y_1 - 1} + \dfrac{320}{10 - y_2}$

où $\{L, U\} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 10 \\ 10 \end{pmatrix} \right\}$

(Svanberg 2) $\overline{\overline{f}}(y; x) = -23 + \dfrac{4}{y_1 - 1} + \dfrac{45}{5 - y_2}$

où $\{L, U\} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\}$


10
5
0
5
10
X_2
X_1
concave
inverse
réelle
linéaire
convexe
svanberg 1
svanberg 2

Annexes au chapitre 2.

Annexe 2.A. : Expression de $\overline{f}(y;x)$.

Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$y \longmapsto f(y)$

Soit le point $x \in \mathbb{R}^n$, point de linéarisation.

Nous allons "linéariser" f autour de x par rapport à

$$\begin{cases} \dfrac{1}{y_j - \ell_j} & \text{si } \dfrac{\partial f}{\partial y_j}\Big|_x < 0 \\ \dfrac{1}{u_j - y_j} & \text{si } \dfrac{\partial f}{\partial y_j}\Big|_x \geq 0 \end{cases}$$

où $\{\ell, u\}$ sont les asymptotes associées à x.

Soit le changement de variable :

$$\underline{t} = (t_j)_{j \in J-} \quad \text{où} \quad J- = \{j \in \{1,\dots,n\} \text{ tq } \frac{\partial f}{\partial y_j}\Big|_x < 0\}$$

$$\text{où} \quad t_j = \frac{1}{y_j - \ell_j}$$

$$\text{et donc} \quad \frac{1}{t_j} = y_j - \ell_j$$

$$y_j = \frac{1}{t_j} + \ell_j$$

$\underline{3} = (3_j)_{j \in J+}$ où $J+ = \{ j \in \{1, \dots, m\} \text{ tq } \frac{\partial f}{\partial y_j}/x \geq 0 \}$

où $\quad 3_j = \frac{1}{v_j - y_j}$

et donc $\quad \frac{1}{3_j} = v_j - y_j$

$$y_j = v_j - \frac{1}{3_j}$$

nous avons alors $f(y) = \hat{f}(\underline{t}, \underline{3})$

linéarisons $\hat{f}(\underline{t}, \underline{3})$ autour de $(\underline{t}^*, \underline{3}^*)$

où $\quad \begin{cases} \underline{t}_j^* = \frac{1}{x_j - 4_j} \\ \underline{3}_j^* = \frac{1}{v_j - x_j} \end{cases}$

nous obtenons au premier ordre :

$$\hat{f}(\underline{t}, \underline{3} ; \underline{t}^*, \underline{3}^*) = \hat{f}(\underline{t}^*, \underline{3}^*) +$$

$$\sum_{j \in J-} \frac{\partial \hat{f}}{\partial t_j}\Big|_{(\underline{t}^*, \underline{3}^*)} (t_j - t_j^*)$$

$$+ \sum_{j \in J+} \frac{\partial \hat{f}}{\partial 3_j}\Big|_{(\underline{t}^*, \underline{3}^*)} (3_j - 3_j^*)$$

Or nous avons :

$$\frac{\partial \varphi}{\partial y_j} = \frac{\partial \hat{\varphi}}{\partial t_j} \quad \frac{\partial t_j}{\partial y_j} \qquad \text{si } j \in J-$$

$$\frac{\partial \varphi}{\partial y_j} = \frac{\partial \hat{\varphi}}{\partial 3_j} \quad \frac{\partial 3_j}{\partial y_j} \qquad \text{si } j \in J+$$

En revenant aux variables de départ :

$$\hat{\varphi}(y ; x) = \varphi(x) +$$

$$\sum_{j \in J-} \frac{\partial \varphi}{\partial y_j}\Big|_x \left(-(x_j - z_j)^2\right) \left[\frac{1}{y_j - z_j} - \frac{1}{x_j - z_j}\right]$$

$$+ \sum_{j \in J+} \frac{\partial \varphi}{\partial y_j}\Big|_x (v_j - x_j)^2 \left[\frac{1}{v_j - y_j} - \frac{1}{v_j - x_j}\right]$$

ou encore :

$$\hat{\varphi}(y ; x) = \varphi(x) +$$

$$\sum_{j \in J-} \left[(x_j - z_j) - \frac{(x_j - z_j)^2}{y_j - z_j}\right] \frac{\partial \varphi}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j)\right] \frac{\partial \varphi}{\partial y_j}\Big|_x$$

Annexe 2.B. : Convexité de $\overline{f}(y;x)$.

Reprenons l'expression de $\overline{f}(y;x)$:

$$\overline{f}(y;x) = f(x) + \sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J-} \left[(x_j - l_j) - \frac{(x_j - l_j)^2}{y_j - l_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

où $\{l, v\}$ est associée à x

$$J+ = \{ j \in \{1 \dots n\} \text{ tq } \frac{\partial f}{\partial y_j}\Big|_x \geq 0 \}$$

$$J- = \{ j \in \{1 \dots n\} \text{ tq } \frac{\partial f}{\partial y_j}\Big|_x < 0 \}$$

Considérons les dérivées premières :

$$\frac{\partial \overline{f}}{\partial y_j} = \begin{cases} \dfrac{(v_j - x_j)^2}{(v_j - y_j)^2} \dfrac{\partial f}{\partial y_j}\Big|_x & \text{si } j \in J+ \\ \dfrac{(x_j - l_j)^2}{(y_j - l_j)^2} \dfrac{\partial f}{\partial y_j}\Big|_x & \text{si } j \in J- \end{cases}$$

Pour les dérivées secondes, nous avons :

$$\frac{\partial^2 \overline{f}}{\partial y_j \partial y_k} = \begin{cases} 0 & \text{si } j \neq k \\ 2\frac{(v_j - x_j)^2}{(v_j - y_j)^3} \frac{\partial f}{\partial y_j}/x & \text{si } j = k \in J+ \\ -2\frac{(x_j - \ell_j)^2}{(y_j - \ell_j)^3} \frac{\partial f}{\partial y_j}/x & \text{si } j = k \in J- \end{cases}$$

On rappelons-nous que nous allons travailler dans l'intervalle $[\underline{x}, \overline{x}] \cap [\ell, v]$.

Dans ce cas $v_j - y_j \geq 0$ et $y_j - \ell_j \geq 0$.

Nous éviterons même le problème de l'annulation du dénominateur en nous restreignant à

$[\underline{x}, \overline{x}] \cap]\ell, v[$. (cfr. Chapitre 3)

Alors $\frac{\partial^2 \overline{f}}{\partial y_j \partial y_k} \geq 0 \quad \forall j, \forall k$

La Hessien de $\overline{f}$ sera défini positif et $\overline{f}$ sera convexe.

Annexe 2.C. : Linéarisation directe.

considérons $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$x \in \mathbb{R}^n$, $\{\ell, v\}$

avec $\quad L \to 0^- \quad -\infty$

$\quad\quad U \to 0^+ \quad +\infty$

Alors :

$$\overline{df}(y;x) = f(x) +$$

$$\lim_{U \to 0^+ \, +\infty} \sum_{j \in J_+} \left[\frac{(u_j - x_j)^2}{u_j - y_j} - (u_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \lim_{L \to 0^- \, -\infty} \sum_{j \in J_-} \left[(x_j - l_j) - \frac{(x_j - l_j)^2}{y_j - l_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$\overline{df}(y;x) = f(x) +$$

$$\sum_{j \in J_+} \lim_{u_j \to 0^+ \, +\infty} \left[\frac{(u_j - x_j)^2}{u_j - y_j} - (u_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J_-} \lim_{l_j \to 0^- \, -\infty} \left[(x_j - l_j) - \frac{(x_j - l_j)^2}{y_j - l_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

Voyons :

$$\lim_{u_j \to 0^+ \, +\infty} \left[\frac{(u_j - x_j)^2}{u_j - y_j} - (u_j - x_j) \right]$$

$$= \lim_{u_j \to 0^+ \, +\infty} \frac{(u_j - x_j)^2 - (u_j - x_j)(u_j - y_j)}{u_j - y_j}$$

$$= \lim_{u_j \to 0^+ \, +\infty} \frac{u_j^2 + x_j^2 - 2u_j x_j - u_j^2 + u_j y_j + x_j u_j - x_j y_j}{u_j - y_j}$$

$$= \lim_{v_j \to 0^+ \infty} \frac{x_j (x_j - y_j) - v_j (x_j - y_j)}{v_j - y_j}$$

$$= \lim_{v_j \to 0^+ \infty} \frac{(x_j - v_j)(x_j - y_j)}{v_j - y_j}$$

C'est une limite du type ∞/∞.

Nous appliquons la règle de l'Hospital :

$$= \lim_{v_j \to 0^+ \infty} -(x_j - y_j) = y_j - x_j$$

Nous avons alors :

$$\lim_{v_j \to 0^+ \infty} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] = y_j - x_j$$

Voyons ensuite :

$$\lim_{z_j \to 0^- \infty} \left[(x_j - z_j) - \frac{(x_j - z_j)^2}{y_j - z_j} \right]$$

$$= \lim_{z_j \to 0^- \infty} \frac{(x_j - z_j)(y_j - z_j) - (x_j - z_j)^2}{y_j - z_j}$$

$$= \lim_{z_j \to 0^- \infty} \frac{x_j y_j - x_j z_j - z_j y_j + z_j^2 - x_j^2 - z_j^2 + 2 z_j x_j}{y_j - z_j}$$

$$= \lim_{z_j \to 0^- \infty} \frac{x_j (y_j - x_j) + z_j (x_j - y_j)}{y_j - z_j}$$

$$= \lim_{z_j \to 0^- \infty} \frac{(x_j - y_j)(z_j - x_j)}{y_j - z_j}$$

C'est une limite du type ∞/∞.

Nous appliquons la règle de l'Hospital :

$$= \lim_{\lambda_j \to 0^-} -(x_j - y_j) = y_j - x_j$$

Nous avons donc :

$$\lim_{\lambda_j \to 0^-} \left[(x_j - \lambda_j) - \frac{(x_j - \lambda_j)^2}{y_j - \lambda_j} \right] = y_j - x_j$$

En résumé, nous pouvons écrire :

$$\overline{f}(y; x) = f(x) +$$

$$\sum_{j \in J_+} (y_j - x_j) \frac{\partial f}{\partial y_j}\Big|_x + \sum_{j \in J_-} (y_j - x_j) \frac{\partial f}{\partial y_j}\Big|_x$$

$$= f(x) + \sum_{j \in \overline{m}} (y_j - x_j) \frac{\partial f}{\partial y_j}\Big|_x$$

nous avons bien obtenu la linéarisation directe.

Annexe 2.D. : Linéarisation convexe.

Considérons $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$x \in \mathbb{R}^n$, $\{\lambda, \mu\}$

avec $\lambda = 0$

$\mu \longrightarrow +\infty$

$$\overline{f}(y;x) = f(x) +$$

$$\lim_{v_j \to 0^+\infty} \sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial f}{\partial y_j}/x$$

$$+ \sum_{j \in J-} \left[x_j - \frac{x_j^2}{y_j} \right] \frac{\partial f}{\partial y_j}/x$$

Or nous avons vu dans l'annexe 2.C. que

$$\lim_{v_j \to 0^+\infty} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] = y_j - x_j$$

Donc

$$\overline{f}(y;x) = f(x) +$$

$$\sum_{j \in J+} (y_j - x_j) \frac{\partial f}{\partial y_j}/x + \sum_{j \in J-} \frac{x_j}{y_j} (y_j - x_j) \frac{\partial f}{\partial y_j}/x$$

nous obtenons ainsi la linéarisation convexe.

Annexe 2.E. : Conservativité :

Soient $\overline{f}(y;x)$ et $\overline{\overline{f}}(y;x)$ 2 approximations de f autour de x, construites sur les asymptotes $\{\underline{L}, \overline{U}\}$ et $\{\overline{L}, \overline{U}\}$

si $\quad \underline{l}_j \le \overline{l}_j < x_j < \overline{\overline{u}}_j \le \overline{u}_j \qquad j = 1 \dots n$

alors $\quad \forall y \in \mathbb{R}^n$ tel que

$$\overline{\overline{l}}_j < y_j < \overline{\overline{u}}_j \qquad j = 1 \dots n$$

nous aurons :

$$\overline{f}(y ; x) \le \overline{\overline{f}}(y ; x)$$

Démontrons cela :

Démontrons donc que $\overline{f}(y;x) - \overline{\overline{f}}(y;x) \le 0$

Par définition :

$$\overline{f}(y;x) - \overline{\overline{f}}(y;x) = f(x) +$$

$$\sum_{j \in J+} \left[\frac{(\overline{u}_j - x_j)^2}{\overline{u}_j - y_j} - (\overline{u}_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J-} \left[(x_j - \overline{l}_j) - \frac{(x_j - \overline{l}_j)^2}{y_j - \overline{l}_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$- f(x)$$

$$- \sum_{j \in J+} \left[\frac{(\overline{\overline{u}}_j - x_j)^2}{\overline{\overline{u}}_j - y_j} - (\overline{\overline{u}}_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$- \sum_{j \in J-} \left[(x_j - \overline{\overline{l}}_j) - \frac{(x_j - \overline{\overline{l}}_j)^2}{y_j - \overline{\overline{l}}_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$\overline{\overline{\rho}}(y;x) - \overline{\overline{\overline{\rho}}}(y;x) =$$

$$\sum_{j\in S+}\left[\frac{(\overline{\overline{v}}_j - x_j)^2}{\overline{\overline{v}}_j - y_j} - (\overline{\overline{v}}_j - x_j) - \frac{(\overline{\overline{\overline{v}}}_j - x_j)^2}{\overline{\overline{\overline{v}}}_j - y_j} + (\overline{\overline{\overline{v}}}_j - x_j)\right]\frac{\partial f}{\partial y_j}\Big|_x$$

$$+\sum_{j\in S-}\left[(x_j - \overline{\overline{z}}_j) - \frac{(x_j - \overline{\overline{z}}_j)^2}{y_j - \overline{\overline{z}}_j} - (x_j - \overline{\overline{\overline{z}}}_j) + \frac{(x_j - \overline{\overline{\overline{z}}}_j)^2}{y_j - \overline{\overline{\overline{z}}}_j}\right]\frac{\partial f}{\partial y_j}\Big|_x$$

Nous allons étudier les signes des termes des deux sommes.

Pour la première somme :

$$\frac{(\overline{\overline{v}}_j - x_j)^2}{\overline{\overline{v}}_j - y_j} - (\overline{\overline{v}}_j - x_j) - \frac{(\overline{\overline{\overline{v}}}_j - x_j)^2}{\overline{\overline{\overline{v}}}_j - y_j} + (\overline{\overline{\overline{v}}}_j - x_j)$$

Nous remarquons que $\begin{cases} \overline{\overline{v}}_j - y_j > 0 \\ \overline{\overline{\overline{v}}}_j - y_j > 0 \end{cases}$

Après réduction au même dénominateur, nous aurons un dénominateur strictement positif.

Etudions le numérateur :

$$(\overline{\overline{v}}_j - x_j)^2(\overline{\overline{\overline{v}}}_j - y_j) - (\overline{\overline{v}}_j - x_j)(\overline{\overline{v}}_j - y_j)(\overline{\overline{\overline{v}}}_j - y_j)$$
$$- (\overline{\overline{\overline{v}}}_j - x_j)^2(\overline{\overline{v}}_j - y_j) + (\overline{\overline{\overline{v}}}_j - x_j)(\overline{\overline{v}}_j - y_j)(\overline{\overline{\overline{v}}}_j - y_j)$$

$$= (\overline{\overline{v}}_j^2 + x_j^2 - 2\overline{\overline{v}}_j x_j)(\overline{\overline{\overline{v}}}_j - y_j) - (\overline{\overline{v}}_j^2 - \overline{\overline{v}}_j y_j - x_j\overline{\overline{v}}_j + x_j y_j)$$
$$(\overline{\overline{\overline{v}}}_j - y_j) - (\overline{\overline{\overline{v}}}_j^2 + x_j^2 - 2\overline{\overline{\overline{v}}}_j x_j)(\overline{\overline{v}}_j - y_j)$$
$$+ (\overline{\overline{\overline{v}}}_j\overline{\overline{v}}_j - \overline{\overline{\overline{v}}}_j y_j - x_j\overline{\overline{v}}_j + x_j y_j)(\overline{\overline{\overline{v}}}_j - y_j)$$

$$= \bar{v}_j^2 \bar{\bar{v}}_j - \bar{v}_j^2 y_j + x_j^2 \bar{\bar{v}}_j - x_j^2 y_j - 2 \bar{v}_j \bar{\bar{v}}_j x_j + 2 \bar{v}_j x_j y_j$$
$$- \bar{v}_j^2 \bar{\bar{v}}_j + \bar{v}_j^2 y_j + \bar{v}_j y_j \bar{\bar{v}}_j - \bar{v}_j y_j^2 + x_j \bar{v}_j \bar{\bar{v}}_j$$
$$- x_j \bar{v}_j y_j - x_j y_j \bar{\bar{v}}_j + x_j y_j^2 - \bar{\bar{v}}_j^2 \bar{v}_j + \bar{\bar{v}}_j^2 y_j - x_j^2 \bar{v}_j$$
$$+ x_j^2 y_j + 2 \bar{\bar{v}}_j x_j \bar{v}_j - 2 \bar{\bar{v}}_j x_j y_j + \bar{\bar{v}}_j^2 \bar{v}_j - \bar{\bar{v}}_j \bar{v}_j y_j$$
$$- \bar{\bar{v}}_j^2 y_j + \bar{\bar{v}}_j y_j^2 - x_j \bar{v}_j \bar{\bar{v}}_j + x_j \bar{v}_j y_j + x_j y_j \bar{\bar{v}}_j - x_j y_j^2$$

$$= x_j^2 \bar{\bar{v}}_j + 2 \bar{v}_j x_j y_j - \bar{v}_j y_j^2 - x_j^2 \bar{v}_j - 2 \bar{\bar{v}}_j x_j y_j + \bar{\bar{v}}_j y_j^2$$

$$= x_j^2 (\bar{\bar{v}}_j - \bar{v}_j) + y_j^2 (\bar{\bar{v}}_j - \bar{v}_j) - 2 x_j y_j (\bar{\bar{v}}_j - \bar{v}_j)$$

$$= (\bar{\bar{v}}_j - \bar{v}_j)(x_j^2 + y_j^2 - 2 x_j y_j) = (\bar{\bar{v}}_j - \bar{v}_j)(x_j - y_j)^2 \leq 0$$

Nous avons ainsi que les termes de la première somme sont tous négatifs. La somme le sera donc aussi.

Pour la seconde somme :

$$(x_j - \bar{z}_j) - \frac{(x_j - \bar{z}_j)^2}{y_j - \bar{z}_j} - (x_j - \bar{\bar{z}}_j) + \frac{(x_j - \bar{\bar{z}}_j)^2}{y_j - \bar{\bar{z}}_j}$$

Nous remarquons que $\begin{cases} y_j - \bar{z}_j > 0 \\ y_j - \bar{\bar{z}}_j > 0 \end{cases}$

Après réduction au même dénominateur, nous aurons un dénominateur strictement positif.

Etudions le numérateur :

$$(x_j - \underline{z}_j)(y_j - \underline{z}_j)(y_j - \overline{\underline{z}}_j) - (x_j - \underline{z}_j)^2 (y_j - \overline{\underline{z}}_j)$$
$$- (x_j - \overline{\underline{z}}_j)(y_j - \underline{z}_j)(y_j - \overline{\underline{z}}_j) + (x_j - \overline{\underline{z}}_j)^2 (y_j - \underline{z}_j)$$

$$= (x_j y_j - x_j \underline{z}_j - \underline{z}_j y_j + \underline{z}_j^2)(y_j - \overline{\underline{z}}_j) - (x_j^2 + \underline{z}_j^2 - 2 x_j \underline{z}_j)$$
$$(y_j - \overline{\underline{z}}_j) - (x_j y_j - x_j \underline{z}_j - \overline{\underline{z}}_j y_j + \overline{\underline{z}}_j \underline{z}_j)(y_j - \overline{\underline{z}}_j)$$
$$+ (x_j^2 + \overline{\underline{z}}_j^2 - 2 x_j \overline{\underline{z}}_j)(y_j - \underline{z}_j)$$

$$= x_j y_j^2 - x_j y_j \overline{\underline{z}}_j - x_j \underline{z}_j y_j + x_j \underline{z}_j \overline{\underline{z}}_j - \underline{z}_j y_j^2 + \underline{z}_j \overline{\underline{z}}_j y_j$$
$$+ \underline{z}_j^2 y_j^2 - \underline{z}_j^2 \overline{\underline{z}}_j - x_j^2 y_j + x_j^2 \overline{\underline{z}}_j - \underline{z}_j^2 y_j + \underline{z}_j^2 \overline{\underline{z}}_j$$
$$+ 2 x_j \underline{z}_j y_j - 2 x_j \underline{z}_j \overline{\underline{z}}_j - x_j y_j^2 + x_j y_j \overline{\underline{z}}_j + x_j \underline{z}_j y_j$$
$$- x_j \underline{z}_j \overline{\underline{z}}_j + \overline{\underline{z}}_j y_j^2 - \overline{\underline{z}}_j^2 y_j - \overline{\underline{z}}_j \underline{z}_j y_j + \overline{\underline{z}}_j^2 \underline{z}_j + x_j^3 y_j$$
$$- x_j^2 \underline{z}_j + \overline{\underline{z}}_j^2 y_j - \overline{\underline{z}}_j^2 \underline{z}_j - 2 x_j \overline{\underline{z}}_j y_j + 2 x_j \overline{\underline{z}}_j \underline{z}_j$$

$$= - \underline{z}_j y_j^2 + x_j^2 \overline{\underline{z}}_j + 2 x_j \underline{z}_j y_j - 2 x_j \overline{\underline{z}}_j y_j + \overline{\underline{z}}_j y_j^2 - x_j^2 \underline{z}_j$$

$$= x_j^2 (\overline{\underline{z}}_j - \underline{z}_j) + y_j^2 (\overline{\underline{z}}_j - \underline{z}_j) - 2 x_j y_j (\overline{\underline{z}}_j - \underline{z}_j)$$

$$= (\overline{\underline{z}}_j - \underline{z}_j)(x_j^2 + y_j^2 - 2 x_j y_j) = (\overline{\underline{z}}_j - \underline{z}_j)(x_j - y_j)^2 \geq 0$$

Nous avons alors que les termes de la seconde somme sont tous négatifs. La somme le sera aussi.
Et donc $\underline{f}(y; x) - \overline{\underline{f}}(y; x) \leq 0$

$$\forall y \text{ tq } \quad \underline{z}_j < y_j < \overline{\underline{v}}_j \qquad j = 1 \dots n$$

Chapitre 3 :

Sous-problème linéarisé.

§ 3.1 : Introduction de "move-limits".

Dans un processus itératif du type de celui que nous étudions, à chaque itération, un sous-problème particulier est traité. Ce sous-problème est, dans la plupart des cas, construit à partir du problème initial en remplaçant les fonctions par des approximations.

Dans ce cas précis, nous avons développé au chapitre 1, une idée nouvelle selon laquelle nous associons à tout point x de $\mathbb{R}^n$, un couple de paramètres $\{L, U\}$, appelés asymptotes mobiles. A travers la description des divers modes de calcul de L et U, nous avons analysé l'intérêt que ces asymptotes pouvaient avoir. Et nous sommes ainsi arrivés à la conclusion qu'elles nous permettent d'agir sur l'intervalle qui borne les variables $[\underline{x}, \overline{x}]$. Ceci peut accélérer la vitesse du processus puisque nous tenons compte de son évolution. Nous en avons déduit qu'il serait intéressant de travailler non dans $[\underline{x}, \overline{x}]$ mais dans l'intersection $[\underline{x}, \overline{x}] \cap [L, U]$.

Si nous voulons continuer dans cette

perspective, nous en arrivons à ce qui suit.

Rappelons-nous l'algorithme :

Pas 0 : Initialisation : - choix du point de départ x^0

- poser $k = 0$

Pas 1 : Soit x^k.

Construction du sous-problème $\overline{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\overline{P}(x^k)$.

Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$

Retourner au pas 1.

A chaque itération, nous possédons un point noté x^k. A ce point x^k, comme vu au chapitre 1, nous associons une paire $\{L^k, U^k\}$.
La méthode utilisée pour la construction de ces asymptotes n'a, pour le moment, aucune importance.

Dans le chapitre 2, nous avons expliqué la façon dont nous construisons une approximation d'une fonction autour d'un point auquel nous avons fait correspondre une paire d'asymptotes.

Il nous est donc possible d'établir les approximations de la fonction objectif f et des contraintes f_i autour de x^k avec $\{L^k, U^k\}$.

Je nous vient alors à l'esprit, l'envie de définir le sous-problème $\overline{P}(x^k)$ comme étant le problème initial où la fonction objectif et les contraintes ont été remplacées par leur approximation autour de x^k et de nous restreindre à travailler sur l'intervalle

$$[\underline{x}, \bar{x}] \cap [L^k, U^k].$$

Cependant, il se pose un problème.

Reprenons l'expression d'une approximation. Par exemple, celle de la fonction objectif f :

$$\overline{f}(y; x^k) = f(x^k) +$$

$$\sum_{j \in J+} \left[\frac{(U_j^k - x_j^k)^2}{U_j^k - y_j} - (U_j^k - x_j^k) \right] \frac{\partial f}{\partial y_j}\Big|_{x^k}$$

$$+ \sum_{j \in J-} \left[(x_j^k - L_j^k) - \frac{(x_j^k - L_j^k)^2}{y_j - L_j^k} \right] \frac{\partial f}{\partial y_j}\Big|_{x^k}$$

où $J+ = \{ j \in \{1, \dots, n\} \text{ tq } \frac{\partial f}{\partial y_j}\Big|_{x^k} \geq 0 \}$

$$J- = \{ j \in \{1, \dots, n\} \text{ tq } \frac{\partial f}{\partial y_j}\Big|_{x^k} < 0 \}$$

Nous avons signalé que nous allons trouver des y appartenant à $[\underline{x}, \bar{x}] \cap [L^k, U^k]$.

Admettons que nous ayons $L_j^k \in [\underline{x}_j, \bar{x}_j]$

pour un $j \in J-$ quelconque. Dans ce cas, y_j peut prendre la valeur ℓ_j^k et il y a alors annulation d'un dénominateur dans $\overline{f}(y; x^k)$ puisque nous y avons $\dfrac{(x_j^k - \ell_j^k)^2}{y_j^k - \ell_j^k}$.

Pour éviter ces inconvénients gênants, nous allons introduire des nouveaux paramètres appelés "move - limits", qui ont pour seul but d'éliminer tout risque d'annulation au dénominateur.

Définissons alors α^k et $\beta^k \in \mathbb{R}^n$ tels que:

$$\left.\begin{array}{l} \ell_j^k < \alpha_j^k \\ \beta_j^k < u_j^k \\ \alpha_j^k < \beta_j^k \end{array}\right\} \quad j = 1 \dots n$$

et travaillons dans $[\underline{x}, \overline{x}] \cap [\alpha^k, \beta^k]$.

Pour que ce nouvel intervalle se rapproche le plus du précédent $[\underline{x}, \overline{x}] \cap [\ell^k, u^k]$, il est utile de choisir, pour tout j, α_j^k dans un petit voisinage de ℓ_j^k et de même pour β_j^k et u_j^k.

K. Svanberg [4] suppère de prendre :

$$\left\{\begin{array}{l} \alpha_j^k = 0.9\ \ell_j^k + 0.1\ x_j^k \\ \beta_j^k = 0.9\ u_j^k + 0.1\ x_j^k \end{array}\right. \quad j = 1 \dots n$$

Mais ceci n'est pas une obligation et toute autre façon de définir explicitement α_j^k et β_j^k peut être prise en considération.

Il est à noter que nous avions imposé d'avoir

$$L_j^k < x_j^k < U_j^k \qquad j = 1 \dots n$$

nous gardons cette contrainte qui, après adaptation, devient :

$$L_j^k < \alpha_j^k < x_j^k < \beta_j^k < U_j^k \qquad j = 1 \dots n$$

Nous sommes à même, maintenant, de formuler l'expression du sous-problème $\overline{P}(x^k)$.

§ 3.2 : Description :

Plaçons-nous à une itération quelconque, soit la $k^{\text{ième}}$ où $k \in \mathbb{N}$.

Soit le point x^k que nous appelons point de linéarisation. Associons-lui par un procédé non spécifié, une paire d'asymptotes $\{L^k, U^k\}$.

Définissons alors $\overline{P}(x^k)$ comme suit : [4]

$$\overline{P}(x^k) \begin{cases} \text{minimiser} & \overline{f}(y ; x^k) \\ \text{s.c.} & \overline{f_i}(y ; x^k) \leq \hat{f}_i \qquad i = 1 \dots m \\ & \max\{\underline{x}_j, \alpha_j^k\} \leq y_j \leq \min\{\beta_j^k, \bar{x}_j\} \\ & \qquad j = 1 \dots n \end{cases}$$

Si nous nous rappelons le problème initial :

$$P \quad \begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots n \end{cases}$$

nous voyons que :

* $\overline{f}(y ; x^k)$ est la "linéarisation de f autour de x^k par rapport à $\frac{1}{y_j - l_j^k}$ et $\frac{1}{u_j^k - y_j}$ "

décrite au chapitre 2.

* $\overline{f}_i(y ; x^k)$ est la "linéarisation de f_i autour de x^k par rapport à $\frac{1}{y_j - l_j^k}$ et $\frac{1}{u_j^k - y_j}$ "

décrite au chapitre 2.

* $(\alpha_j^k)_j$ et $(\beta_j^k)_j$ sont les "move-limits" associées à l^k et u^k et décrites ci-dessus.

L'intervalle

$$[\max\{\underline{x}_j, \alpha_j^k\}, \min\{\beta_j^k, \bar{x}_j\}]$$

exprime bien notre intention de nous restreindre à $[\underline{x}, \bar{x}] \cap [\alpha^k, \beta^k]$

Chapitre 4 :

Convergence de l'algorithme de Svanberg.

§ 4.1 : Introduction.

La méthode des asymptotes mobiles est relativement récente puisqu'elle a été présentée en mai 1986 par K. Svanberg du "Royal Institue of Technology" à Stockholm [4].

Jusqu'à présent, aucune étude n'a porté sur la convergence d'une telle méthode. C'est pourquoi, il nous a semblé intéressant de l'analyser plus profondément afin de pouvoir établir certains résultats à ce propos.

Voici résumés, les principes de base et les hypothèses avec lesquels nous avons travaillé.

Le problème considéré est le suivant :

$$P \quad \begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots n \end{cases}$$

où $\begin{cases} f : \mathbb{R}^n \longrightarrow \mathbb{R} \\ f_i : \mathbb{R}^n \longrightarrow \mathbb{R} \qquad i = 1 \dots m \end{cases}$

La résolution du problème non linéaire P

utilise un algorithme itératif du type :

Pas 0 : Initialisation : - choix du point de départ x^0
- poser $k = 0$

Pas 1 : Soit x^k.
Construction du sous-problème $\overline{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\overline{P}(x^k)$.
Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$
Retourner au pas 1.

La particularité de la méthode réside en la manière dont sont construits les sous-problèmes $\overline{P}(x^k)$:

$$\overline{P}(x^k) \begin{cases} \text{minimiser} & \overline{f}(y ; x^k) \\ \text{s.c.} & \overline{f_i}(y ; x^k) \leq \hat{f_i} \quad i = 1 \dots m \\ & \max\{\underline{x}_j, \alpha_j^k\} \leq y_j \leq \min\{\beta_j^k, \overline{x}_j\} \\ & j = 1 \dots n \end{cases}$$

Les fonctions $\overline{f}(y ; x^k)$ et $\overline{f_i}(y ; x^k)$ sont les linéarisations par rapport aux variables réciproques autour de x^k proposées par Svanberg, des fonctions $f(x)$ et $f_i(x)$ du problème P

original.

Il reste à préciser la façon dont sont choisies les asymptotes L et U ainsi que les "move-limits". Les asymptotes mobiles ont été fixées comme suit :

$$\begin{cases} L_j^k = x_j^k - (\bar{x}_j - \underline{x}_j) \\ U_j^k = x_j^k + (\bar{x}_j - \underline{x}_j) \end{cases}$$

où $k \in \mathbb{N}$, $j = 1 \dots n$

tandis que les "move-limits" sont telles que :

$$\begin{cases} \alpha_j^k = 0.9 \; L_j^k + 0.1 \; x_j^k \\ \beta_j^k = 0.9 \; U_j^k + 0.1 \; x_j^k \end{cases}$$

où $k \in \mathbb{N}$, $j = 1 \dots n$

Ces paramètres vérifient alors les inégalités :

$$L_j^k < \alpha_j^k < x_j^k < \beta_j^k < U_j^k$$

Ayant ainsi exposé le cadre de travail, nous pouvons aborder la convergence en elle-même.

§ 4.2 : Convergence sans recherche linéaire.

Tout d'abord, la méthode a été étudiée sans avoir été modifiée. En posant certaines conditions sur les fonctions, un premier

résultat de convergence a pu être établi.

Théorème 2 :

Sous les hypothèses suivantes :

- la fonction objectif f est concave et continûment différentiable.
- les contraintes f_i $(i = 1 \dots m)$ sont concaves et continûment différentiables.
- $\forall x$, $\overline{P}(x)$ vérifie la condition de Slater
- l'ensemble

  $E_0 = \{ x \text{ admissibles tels que } f(x) \leq f(x^0) \}$

  est un compact.

alors

tout point d'accumulation de la suite (x^k) des itérés obtenus par l'algorithme, est un point de Kuhn - Tucker du problème P.

La démonstration se base sur un théorème de Zangwill [5] et utilise certains résultats présentés sous forme de lemmes.

4.2.1 : Lemmes.

Lemme 1 :

Si les fonctions f_i $i = 1 \dots m$ sont concaves et continûment différentiables, alors le domaine "linéarisé" en x^k défini par:

$$\begin{cases} \overline{f_i}\,(y ; x^k) \leq \hat{f}_i & i = 1 \dots m \\ \max \{ \alpha_j{}^k , \underline{x}_j \} \leq y_j \leq \min \{ \beta_j{}^k , \bar{x}_j \} & j = 1 \dots n \end{cases}$$

est inclus dans le domaine initial défini par:

$$\begin{cases} f_i\,(y) \leq \hat{f}_i & i = 1 \dots m \\ \underline{x}_j \leq y_j \leq \bar{x}_j & j = 1 \dots n \end{cases}$$

et cela, quel que soit le point de linéarisation x^k.

Démonstration du Lemme 1 :

Soit y appartenant au domaine "linéarisé". Montrons que y appartient au domaine initial.

a) Montrons tout d'abord que y vérifie

$$f_i(y) \leq \hat{f}_i \qquad i = 1 \dots m$$

Par hypothèse y vérifie $\overline{f}_i(y ; x^k) \leq \hat{f}_i \quad i = 1 \dots m$

Il suffit alors de montrer que

$$\overline{f}_i(y ; x^k) \geq f_i(y)$$

ou bien $f_i(y) - \overline{f}_i(y ; x^k) \leq 0 \qquad i = 1 \dots m$

or f_i étant concave, nous avons :

$$f_i(y) \leq f_i(x^k) + \sum_{j=1}^{n} (y_j - x_j^k) \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}$$

Nous connaissons aussi l'expression de $\overline{f}_i(y ; x^k)$:

$$\overline{f}_i(y ; x^k) = f_i(x^k) +$$

$$\sum_{j \in J+} \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}$$

$$+ \sum_{j \in J-} \left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}$$

où $\begin{cases} J+ = \{ j \in \{1, \dots, n\} \text{ tq } \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k} \geq 0 \} \\ J- = \{ j \in \{1, \dots, n\} \text{ tq } \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k} < 0 \} \end{cases}$

Utilisant ces deux résultats, nous pouvons écrire :

$$f_i(y) - \overline{f_i}(y; x^k) \leq$$

$$\sum_{j \in J+} \left[(y_j - x_j^k) - \frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} + (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

$$+ \sum_{j \in J-} \left[(y_j - x_j^k) - (x_j^k - \ell_j^k) + \frac{(x_j^k - \ell_j^k)^2}{y_j - \ell_j^k} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Etudions le second membre terme à terme.

Pour le premier terme :

$$\sum_{j \in J+} \left[(y_j - x_j^k) - \frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} + (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Etudions plus particulièrement un terme de la somme :

$$\left[(y_j - x_j^k) - \frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} + (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Nous savons que $\frac{\partial f_i}{\partial y_j}\Big|_{x^k} \geq 0$ car $j \in J+$.

Le signe de ce terme dépend alors du facteur qui précède.

Réduisons le tout au même dénominateur :

$$\frac{(y_j - x_j^k)(v_j^k - y_j) - (v_j^k - x_j^k)^2 + (v_j^k - x_j^k)(v_j^k - y_j)}{v_j^k - y_j}$$

Or, par hypothèse, y_j vérifie $y_j \leq \min\{\beta_j^k, \bar{x}_j\}$

et β_j^k vérifie $\beta_j^k < v_j^k$.
Donc $y_j \leq \beta_j^k < v_j^k$ et $v_j^k - y_j > 0$.

Quant au numérateur :

$$y_j v_j^k - y_j^2 - x_j^k v_j^k + x_j^k y_j - (v_j^k)^2 - (x_j^k)^2 + 2 v_j^k x_j^k + (v_j^k)^2 - v_j^k y_j - x_j^k v_j^k + x_j^k y_j$$
$$= -(y_j - x_j^k)^2 \leq 0$$

Le terme étudié est alors négatif.
Comme ceci est valable quel que soit j appartenant à $J+$, le premier terme est négatif.
De même, étudions le second terme :

$$\sum_{j \in J-} \left[(y_j - x_j^k) - (x_j^k - z_j^k) + \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Un terme de cette somme sera :

$$\left[(y_j - x_j^k) - (x_j^k - z_j^k) + \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Nous savons que $\frac{\partial f_i}{\partial y_j}\Big|_{x^k} < 0$ car $j \in J-$.

Etudions le signe de l'autre facteur :
Réduisons-le au même dénominateur :

$$\frac{(y_j - x_j^k)(y_j - z_j^k) - (x_j^k - z_j^k)(y_j - z_j^k) + (x_j^k - z_j^k)^2}{y_j - z_j^k}$$

or, par hypothèse, y_j vérifie $y_j \geq \max\{\underline{x}_j, a_j^k\}$
et a_j^k vérifie $a_j^k > \ell_j^k$.

Donc $\ell_j^k < a_j^k \leq y_j$ et $y_j - \ell_j^k > 0$

quant au numérateur :

$$y_j^2 - y_j \ell_j^k - x_j^k y_j + x_j^k \ell_j^k - x_j^k y_j + x_j^k \ell_j^k + \ell_j^k y_j - (\ell_j^k)^2 + (x_j^k)^2 + (\ell_j^k)^2 - 2 x_j^k \ell_j^k$$
$$= (y_j - x_j^k)^2 \geq 0$$

Ce terme est donc ≤ 0.
Comme ceci est valable quel que soit $j \in J-$,
le second terme sera négatif.

Nous déduisons alors

$$f_i(y) - \overline{f_i}(y; x^k) \leq 0$$

Le résultat est valable $\forall i$ $i = 1 \dots m$ et $\forall x^k$ puisqu'aucune restriction ne fut posée sur leur choix.

b) Montrons ensuite que y vérifie :

$$\underline{x}_j \leq y_j \leq \bar{x}_j \qquad j = 1 \dots n$$

or, par hypothèse, y_j vérifie

$$\max \{ \alpha_j^k , \underline{x}_j \} \leq y_j \leq \min \{ \beta_j^k , \bar{x}_j \}$$

Donc $y_j \leq \bar{x}_j$ et $y_j \geq \underline{x}_j$

et $\underline{x}_j \leq y_j \leq \bar{x}_j \qquad j = 1 \dots n$

C Q F D

Lemme 2 :

Si x^* est optimum pour $\bar{P}(x^*)$, alors x^* est un point de Kuhn - Tucker pour P lorsque $\bar{P}(x^*)$ vérifie une condition de qualification de contraintes (par exemple : celle de Slater).

Démonstration du lemme 2 :

x^* est optimum pour $\bar{P}(x^*)$.

Comme $\bar{P}(x^*)$ vérifie une condition de qualification de contraintes, x^* est point de Kuhn et Tucker de $\bar{P}(x^*)$.

Donc, nous pouvons écrire :

$$\exists \lambda \in \mathbb{R}^m \quad , \quad \exists v, w \in \mathbb{R}^n \quad tq$$

$$\begin{cases} \nabla \bar{f}(x^*; x^*) + \sum_{i=1}^{m} \lambda_i \nabla \bar{f}_i(x^*; x^*) \\ - \sum_{j=1}^{n} v_j e_j + \sum_{j=1}^{n} w_j e_j = 0 \end{cases}$$

$$\begin{cases} \lambda_i \left(\overline{f}_i(x^*; x^*) - \hat{f}_i \right) = 0 & i = 1 \dots m \\ \nu_j \left(\max \{ \underline{x}_j , \alpha_j^* \} - x_j^* \right) = 0 \\ \nu_j \left(x_j^* - \min \{ \overline{x}_j , \beta_j^* \} \right) = 0 & j = 1 \dots n \end{cases}$$

Dans le chapitre 2, nous avons établi que :

$$\nabla \overline{f}(x^*; x^*) = \nabla f(x^*)$$

$$\nabla \overline{f}_i(x^*; x^*) = \nabla f_i(x^*) \qquad i = 1 \dots m$$

Nous avons alors :

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x^*) - \sum_{j=1}^{n} \nu_j e_j + \sum_{j=1}^{n} \nu_j e_j = 0$$

De même, au chapitre 2, nous avons vu que :

$$\overline{f}_i(x^*; x^*) = f_i(x^*) \qquad i = 1 \dots m$$

Nous en déduisons :

$$\lambda_i \left(f_i(x^*) - \hat{f}_i \right) = 0 \qquad i = 1 \dots m$$

Etudions la troisième égalité :

$$\nu_j \left(\max \{ \underline{x}_j , \alpha_j^* \} - x_j^* \right) = 0$$

Si $\nu_j = 0$, ceci nous permet d'affirmer :

$$\nu_j \left(\underline{x}_j - x_j^* \right) = 0$$

Si $\nu_j \neq 0$, deux cas se présentent :

1) $\max \{ \underline{x}_j , \alpha_j^* \} = \underline{x}_j$

alors $\max \{ \underline{x}_j , \alpha_j^* \} - x_j^* = 0$

implique $\underline{x}_j = x_j^*$

ce qui nous donne $\nu_j \left(\underline{x}_j - x_j^* \right) = 0$

e) $\max \{ \underline{x}_j , \alpha_j^* \} = \alpha_j^*$

alors $\max \{ \underline{x}_j , \alpha_j^* \} - x_j^* = 0$

implique $\alpha_j^* = x_j^*$

or, par hypothèse, α_j^* vérifie :

$$\ell_j^* < \alpha_j^* < x_j^* < \beta_j^* < u_j^* .$$

Ce cas est donc impossible.

Dans tous les cas possibles, nous avons donc pu établir $\quad \nu_j \ (\underline{x}_j - x_j^*) = 0 \qquad j = 1 \dots n$

Un raisonnement similaire peut s'appliquer à la quatrième égalité.

Il nous donne : $\quad \omega_j \ (x_j^* - \bar{x}_j) = 0 \quad j = 1 \dots n$

Nous avons alors :

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x^*) - \sum_{j=1}^{n} \nu_j e_j + \sum_{j=1}^{n} \omega_j e_j = 0 \\ \lambda_i \ (f_i(x^*) - \hat{f}_i) = 0 \qquad i = 1 \dots m \\ \nu_j \ (\underline{x}_j - x_j^*) = 0 \\ \omega_j \ (x_j^* - \bar{x}_j) = 0 \end{cases} \qquad j = 1 \dots n$$

Nous avons ainsi montré que x^* est point de Kuhn et Tucker du problème P.

CQFD

Lemme 3 :

Soit un point x admissible pour P, n'étant pas point de Kuhn - Tucker pour P.
Soit y, l'optimum de $\overline{P}(x)$ qui vérifie la condition de Slater.

Nous avons :

$(y - x)$ est une direction de descente pour la fonction objectif f au point x

i.e. $\quad < \nabla f(x) , y - x > \quad < 0$

Démonstration du Lemme 3 :

x est un point admissible pour P, donc

$$f_i(x) \leq \hat{f}_i \qquad i = 1 \dots m$$

Or, dans le chapitre 2, nous avons vu que

$$\overline{f_i}(x ; x) = f_i(x) \qquad i = 1 \dots m$$

Donc, nous avons $\overline{f_i}(x ; x) \leq \hat{f}_i \qquad i = 1 \dots m$

De plus, x admissible pour P nous donne :

$$\underline{x}_j \leq x_j \leq \overline{x}_j$$

et par hypothèse, α_j et β_j sont tels que

$$\alpha_j < x_j < \beta_j$$

Donc, nous avons bien :

$$\max \{ \underline{x}_j , \alpha_j \} \leq x_j \leq \min \{ \beta_j , \overline{x}_j \} \qquad j = 1 \dots n$$

x est alors point admissible pour $\overline{P}(x)$.

Comme y est optimum pour $\overline{P}(x)$, nous pouvons écrire : $\overline{f}(y;x) \leq \overline{f}(x;x)$
De plus, si $\overline{f}(y;x) = \overline{f}(x;x)$, x serait optimum pour $\overline{P}(x)$. Par le lemme 2, nous obtenons que x est point de Kuhn-Tucker pour P. Ce qui contredit les hypothèses.
On aura alors $\overline{f}(y;x) < \overline{f}(x;x)$
Dans le chapitre 2, nous avons encore établi que $\overline{f}$ est une fonction convexe. Cela nous permet de déduire :

$$0 > \overline{f}(y;x) - \overline{f}(x;x) \geq \langle \nabla \overline{f}(x;x), y-x \rangle$$

Et comme $\nabla \overline{f}(x;x) = \nabla f(x)$ (cf chap 2)
nous avons finalement :

$$\langle \nabla f(x), (y-x) \rangle < 0$$

CQFD

Lemme 4 :

Si nous avons une suite $(x^k) \to x$
alors à partir d'un certain rang
$\frac{\partial f_i}{\partial y_j}\Big|_x$ est de même signe que $\frac{\partial f_i}{\partial y_j}\Big|_{x^k}$
où f_i est supposée appartenir à C^1.

Démonstration du lemme 4 :

Montrons qu'il existe k_0 tel que $\forall k \geq k_0$

$$\frac{\partial f_i}{\partial y_j}\Big|_{x^k} \cdot \frac{\partial f_i}{\partial y_j}\Big|_{x} \geq 0$$

Par l'absurde, supposons le contraire :

$$\forall k \quad \exists n_k \geq k \text{ tel que } \frac{\partial f_i}{\partial y_j}\Big|_{x^{n_k}} \cdot \frac{\partial f_i}{\partial y_j}\Big|_{x} < 0$$

Si $\frac{\partial f_i}{\partial y_j}\Big|_{x} = 0$ ceci est absurde

si $\frac{\partial f_i}{\partial y_j}\Big|_{x} < 0$ alors

$\forall k \quad \exists n_k \geq k$ tel que $\frac{\partial f_i}{\partial y_j}\Big|_{x^{n_k}} > 0$

Or nous savons que $\frac{\partial f_i}{\partial y_j}\Big|_{x^{n_k}} \longrightarrow_0 \frac{\partial f_i}{\partial y_j}\Big|_{x}$

car $(x^k) \longrightarrow_0 x$ et $f_i \in C^2$.

Or nous savons aussi que :

$$\frac{\partial f_i}{\partial y_j}\Big|_{x^{n_k}} > 0 \quad \forall k \quad \text{et} \quad \frac{\partial f_i}{\partial y_j}\Big|_{x} < 0$$

Cette situation est alors absurde.

si $\frac{\partial f_i}{\partial y_j}\Big|_x > 0$ nous pouvons montrer de façon semblable que l'hypothèse est absurde.

CQFD

4.2.2 : Démonstration du théorème 1.

Nous allons maintenant démontrer le théorème de convergence énoncé plus haut en nous basant sur les lemmes qui précèdent.

Sous les hypothèses suivantes :

- la fonction objectif f est concave et continûment différentiable.
- les contraintes f_i $(i = 1 \dots m)$ sont concaves et continûment différentiables.
- $\forall x$, $\overline{F}(x)$ vérifie la condition de Slater.
- l'ensemble

  $E_0 \equiv \{ x \text{ admissibles tels que } f(x) \leq f(x^0) \}$

  est un compact.

alors tout point d'accumulation de la suite (x^k) des itérés obtenus par l'algorithme, est un point de Kuhn-Tucker du problème P.

Démonstration :

Nous allons nous baser sur le théorème de Zangwill [5] :

- Soit un problème d'optimisation sur X et Ω l'ensemble des points satisfaisant une certaine condition nécessaire d'optimalité.

- Soit $A : X \longrightarrow Y$ une application multivoque et soit la suite (x^k) donnée par l'algorithme i.e. $x^{k+1} \in A(x^k)$

Si les 3 conditions suivantes sont vérifiées :

- C_1 : les points x^k sont tous contenus dans un compact $K \subset X$
- C_2 : il existe une fonction de descente z i.e. z continue et $\forall\, x \in X$
  - i) $x \notin \Omega \quad z(y) < z(x) \quad \forall y \in A(x)$
  - ii) $x \in \Omega \quad z(y) \leq z(x) \quad \forall y \in A(x)$
- C_3 : l'application multivoque A est fermée sur X / Ω et $\forall x \in X / \Omega \quad A(x) \neq \phi$

alors

tout point d'accumulation de la suite (x^k) est un point de Ω.

Nous allons considérer ici :

$X \equiv$ l'ensemble des points admissibles pour P.

$A : X \multimap Y \equiv \mathcal{P}(X)$: application multivoque qui, à un point admissible pour P, fait correspondre les solutions optimales de $\overline{P}(x)$.

$\mathcal{S} \equiv$ l'ensemble des points de Kuhn-Tucker de P.

Montrons tout d'abord, que la définition de A est connexe.

$\forall x \in X$, nous savons, par hypothèse, que le problème $\overline{P}(x)$ admettra au moins un point admissible et donc, au moins une solution optimale qui étant admissible pour $\overline{P}(x)$, le sera aussi pour P par le lemme 1.
La définition de A est donc connexe.

Considérons le cas où x^0 est admissible pour P :

Dans ce cas, $\forall k$ x^k appartient au domaine admissible de $\overline{P}(x^{k-1})$ et donc, par le lemme 1, $x^k \in X$.
La suite (x^k) est donc incluse dans X.

Il reste à vérifier les 3 conditions de Zangwill.

Vérification de la condition C_1 :

La condition C_2 (vérifiée ci-dessous) nous donne :

$$\forall k \quad f(x^k) \geq f(x^{k+1})$$

Donc $\forall k$

$x^k \in \{ x \in X \text{ tels que } f(x) \leq f(x^0) \}$

qui est un ensemble compact par hypothèse.

Vérification de la condition C_2 :

La fonction objectif f satisfait à la définition de fonction de descente

i.e. f continue et $\forall x \in X$, $\forall y \in A(x)$

i) $x \notin \Omega$ $\quad f(y) < f(x)$

ii) $x \in \Omega$ $\quad f(y) \leq f(x)$

Démontrons ii) :

Si $x \in \Omega$, x est point de Kuhn-Tucker pour P. Alors x est aussi point de Kuhn-Tucker pour $\overline{P}(x)$.

En effet, nous savons que :

$$\begin{cases} \exists\, \lambda \in \mathbb{R}^m \quad , \exists\, \pi, \omega \in \mathbb{R}^n \quad \text{tels que :} \\ \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) - \sum_{j=1}^{n} \pi_j e_j + \sum_{j=1}^{n} \omega_j e_j = 0 \\ \lambda_i \quad (f_i(x) - \hat{f}_i) = 0 \qquad i = 1 \dots m \\ \pi_j \quad (\underline{x}_j - x_j) = 0 \\ \omega_j \quad (x_j - \bar{x}_j) = 0 \qquad j = 1 \dots n \end{cases}$$

Par le chapitre 2, nous savons que :

$$\nabla f(x) = \nabla \overline{f}(x ; x)$$

$$\nabla f_i(x) = \nabla \overline{f}_i(x ; x) \qquad i = 1 \dots m$$

$$f_i(x) = \overline{f}_i(x ; x) \qquad i = 1 \dots m$$

Nous avons alors :

$$\begin{cases} \nabla \overline{f}(x ; x) + \sum_{i=1}^{m} \lambda_i \nabla \overline{f}_i(x ; x) \\ \qquad - \sum_{j=1}^{n} \pi_j e_j + \sum_{j=1}^{n} \omega_j e_j = 0 \\ \lambda_i \quad (\overline{f}_i(x ; x) - \hat{f}_i) = 0 \qquad i = 1 \dots m \end{cases}$$

De plus, $\pi_j (\underline{x}_j - x_j) = 0$

Si $\pi_j = 0$, nous avons aussi que :

$$\pi_j (\max \{\underline{x}_j, \alpha_j\} - x_j) = 0$$

Si $\pi_j \neq 0$, nous déduisons $\underline{x}_j = x_j$.

Or x est admissible pour $\overline{P}(x)$ puisque x est admissible pour P (ceci a été démontré dans le lemme 3).

Donc $\max \{\underline{x}_j, \alpha_j\} \leq x_j = \underline{x}_j$

et $\max \{\underline{x}_j, \alpha_j\} \geq \underline{x}_j$

Donc $\max \{ \underline{x}_j , a_j \} = \underline{x}_j$.
Ce qui nous permet d'écrire :

$$v_j (\max \{ \underline{x}_j , a_j \} - x_j) = 0$$

Et ceci $\forall j = 1 \dots n$.

Un raisonnement semblable nous conduire à déduire que

$$w_j (x_j - \min \{ \beta_j , \bar{x}_j \}) = 0$$

x sera donc bien point de Kuhn - Tucker pour $\overline{P}(x)$ puisque :

$$\begin{cases} \exists \lambda \in \mathbb{R}^m , \quad \exists v, w \in \mathbb{R}^n \text{ tels que :} \\ \nabla \overline{f}(x;x) + \sum_{i=1}^{m} \lambda_i \nabla \overline{f_i}(x;x) \\ \qquad - \sum_{j=1}^{n} v_j e_j + \sum_{j=1}^{n} w_j e_j = 0 \\ \lambda_i (\overline{f_i}(x;x) - \hat{f_i}) = 0 \qquad i = 1 \dots m \\ v_j (\max \{ \underline{x}_j , \alpha_j \} - x_j) = 0 \\ w_j (x_j - \min \{ \beta_j , \bar{x}_j \}) = 0 \qquad j = 1 \dots n \end{cases}$$

Comme $\overline{P}(x)$ vérifie la condition de Slater ,
x est optimum pour $\overline{P}(x)$

i.e. $\overline{f}(y;x) \geq \overline{f}(x;x) \quad y \in A(x)$

Comme , par hypothèse , y est optimal pour $\overline{P}(x)$:

$$\overline{f}(x;x) \geq \overline{f}(y;x)$$

Donc $\overline{f}(y;x) = \overline{f}(x;x)$.

Et, par le chapitre 2, $\overline{f}(x;x) = f(x)$.

Et, par le lemme 1 et f concave :

$$\overline{f}(y;x) \geq f(y)$$

Donc, nous avons finalement :

$$f(y) \leq f(x)$$

Démontrons i) :

Si $x \in \Omega$, x n'est pas point de Kuhn-Tucker pour P.

Deux cas sont alors possibles :

① x est point de Kuhn-Tucker pour $\overline{P}(x)$

② x n'est pas point de Kuhn-Tucker pour $\overline{P}(x)$.

Traitons le cas ② :

x n'est alors pas optimum pour $\overline{P}(x)$

donc $\overline{f}(x;x) > \overline{f}(y;x) \quad \forall y \in A(x)$

i.e. $\overline{f}(y;x) - \overline{f}(x;x) < 0$

Par le chapitre 2, $\overline{f}$ est convexe et

$$\nabla \overline{f}(x;x) = \nabla f(x)$$

Donc

$$0 > \overline{f}(y;x) - \overline{f}(x;x)$$

$$\geq < \nabla \overline{f}(x;x), y - x >$$

i.e. $0 > < \nabla f(x), y - x >$

Comme f est concave :

$$0 > \langle \nabla f(x), y - x \rangle \geq f(y) - f(x)$$

i.e. $f(y) < f(x)$

Traitons le cas ② :

x est alors optimum pour $\bar{P}(x)$.

Par le lemme 2, x est alors point de Kuhn-Tucker pour P.

Or $x \notin \mathcal{D}$, donc ce cas est impossible.

Vérification de la condition C_3 :

Soit une suite $(x^k) \longrightarrow x$

$(y^k) \longrightarrow y$

où $y^k \in A(x^k) \quad \forall k$.

A-t-on $y \in A(x)$?

1) Montrons que y est admissible pour $\bar{P}(x)$:

- a-t-on $\overline{f_i}(y ; x) \leq \hat{f}_i$? $\quad i = 1 \dots m$

Or nous savons que :

si $\begin{cases} J+ = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_x \geq 0 \} \\ J- = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_x < 0 \} \end{cases}$

alors $\overline{f}_i(y;x) = f(x) +$

$$\sum_{j\in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j)\right] \frac{\partial f_i}{\partial y_j}\Big|_x$$

$$+ \sum_{j\in J-} \left[(x_j - z_j) - \frac{(x_j - z_j)^2}{y_j - z_j}\right] \frac{\partial f_i}{\partial y_j}\Big|_x$$

Nous savons que y^k est admissible pour $\overline{P}(x^k)$ puisque $y^k \in A(x^k)$.

Donc $\overline{f}_i(y^k; x^k) \leq \hat{f}_i \qquad i = 1 \dots m$

ou bien

$$f_i(x^k) + \sum_{j\in J'+} \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j^k} - (v_j^k - x_j^k)\right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

$$+ \sum_{j\in J'-} \left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j^k - z_j^k}\right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \leq \hat{f}_i$$

où $\begin{cases} J'+ = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} \geq 0 \} \\ J'- = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} < 0 \} \end{cases}$

Il suffit alors de passer à la limite pour $k \to +\infty$.

Par le Lemme 4, les sommes sur $J+$ et $J'+$ se feront sur les mêmes indices (ainsi que sur

J- et J'-), à partir d'un certain rang.

De plus $x^k \longrightarrow x$

$y^k \longrightarrow y$

Et, par définition, nous avons :

$$\lim_{k \to +\infty} U_j^k = \lim_{k \to +\infty} x_j^k + (\bar{x}_j - \underline{x}_j) = U_j$$

$$\lim_{k \to +\infty} L_j^k = \lim_{k \to +\infty} x_j^k - (\bar{x}_j - \underline{x}_j) = L_j$$

Nous obtiendrons alors :

$$\overline{P}_i(y; x) \leq \hat{P}_i \qquad \forall i = 1 \dots m$$

• a-t-on $\max \{\underline{x}_j, \alpha_j\} \leq y_j \leq \min_{j=1\dots n} \{\beta_j, \bar{x}_j\}$?

Comme y^k est admissible pour $\overline{P}(x^k)$:

$$\max \{\underline{x}_j, \alpha_j^k\} \leq y_j^k \leq \min_{j=1\dots n} \{\beta_j^k, \bar{x}_j\}$$

Comme, de plus,

$$\lim_{k \to +\infty} \alpha_j^k = 0.9 \lim_{k \to +\infty} L_j^k + 0.1 \lim_{k \to +\infty} x_j^k$$

$$= \alpha_j$$

$$\lim_{k \to +\infty} \beta_j^k = 0.9 \lim_{k \to +\infty} U_j^k + 0.1 \lim_{k \to +\infty} x_j^k$$

$$= \beta_j$$

Nous aurons donc bien le résultat.

Car $\forall k \quad y_j^k \geq \underline{x}_j$ et $y_j^k \geq a_j^k$.

Nous passons à la limite sur $k \to +\infty$:

$$y_j \geq \underline{x}_j \quad \text{et} \quad y_j \geq a_j .$$

Donc $y_j \geq \max \{ \underline{x}_j , a_j \}$

Et de même pour $y_j \leq \min \{ \beta_j , \bar{x}_j \}$.

Nous avons ainsi montré que y est admissible pour $\bar{P}(x)$.

2) Montrons que y est optimum pour $\bar{P}(x)$:

i.e. $\bar{f}(z ; x) \geq \bar{f}(y ; x)$

$\forall z$ admissible pour $\bar{P}(x)$

① Si $\forall i = 1 \dots m \quad \bar{f}_i(z ; x) - \hat{f}_i = -\varepsilon_i < 0$

alors z est admissible pour $\bar{P}(x^k)$ pour k suffisamment grand.

En effet,

$\exists k_0$ tel que $\forall k \geq k_0 \quad \bar{f}_i(z ; x^k) \leq \hat{f}_i$

sinon nous aurions :

$\forall k \quad \exists n_k \geq k$ tel que $\bar{f}_i(z ; x^{n_k}) > \hat{f}_i$

Considérons la sous-suite $\left(\bar{f}_i(z ; x^{n_k}) - \hat{f}_i \right)_k$

Nous aurons

$$\left(\overline{f_i}(z ; x^{n_k}) - \hat{f_i} \right) \longrightarrow 0 \left(\overline{f_i}(z ; x) - \hat{f_i} \right)$$

et donc :

$$\left(\overline{f_i}(z ; x^{n_k}) - \hat{f_i} \right) \longrightarrow 0 - \varepsilon_i$$

Ce qui est absurde puisque la sous-suite est choisie de telle sorte que :

$$\forall k \quad \overline{f_i}(z ; x^{n_k}) - \hat{f_i} > 0$$

Il nous faut encore montrer que z vérifie :

$$\max \{ \underline{x}_j , \alpha_j^k \} \leq z_j \leq \min \{ \beta_j^k , \bar{x}_j \} \quad j = 1 \ldots n$$

Montrons tout d'abord la première inégalité :

$$\max \{ \underline{x}_j , \alpha_j^k \} \leq z_j$$

Deux cas se présentent :

1. Si $\max \{ \underline{x}_j , \alpha_j^k \} = \underline{x}_j \qquad j \in J^1$

   Dans ce cas, comme z est admissible pour $\overline{P}(x)$ $\quad z_j \geq \underline{x}_j$

2. Si $\max \{ \underline{x}_j , \alpha_j^k \} = \alpha_j^k \qquad j \in J^2$

   a-t-on $\alpha_j^k \leq z_j$?

* Si $\forall j \in J^2$, nous avons $\alpha_j < z_j$ alors pour k suffisamment grand, nous aurons $\alpha_j^k \leq z_j$.

Sinon

$\forall k \quad \exists m_k$ tel que $a_j^{m_k} > z_j$.

Or $(a_j^{m_k})_k \longrightarrow a_j \geq z_j$

Comme nous avons supposé $a_j < z_j$, cette hypothèse est absurde et nous obtenons donc bien $a_j^k \leq z_j$.

* Si $\exists j \in J^2$ tel que $a_j = z_j$

Par hypothèse , a_j est choisi tel que $a_j < x_j$.

Soit $z_{jm} = \left(1 - \frac{1}{m}\right) z_j + \frac{1}{m} x_j$.

Nous observons que $(z_{jm})_m \longrightarrow z_j$

et $a_j - z_{jm} = a_j - \left(1 - \frac{1}{m}\right) z_j - \frac{1}{m} x_j$

$$= a_j - z_j + \frac{1}{m}(z_j - x_j)$$

$$= a_j - z_j + \frac{1}{m}(a_j - x_j) - \frac{1}{m}(a_j - z_j)$$

$$= \underbrace{(a_j - z_j)}_{\leq 0}\left(1 - \frac{1}{m}\right) + \frac{1}{m}\underbrace{(a_j - x_j)}_{< 0}$$

$$< 0$$

Et donc $a_j < z_{jm} \quad \forall j \in J^2$

Nous nous retrouvons alors avec z_{jm} dans le cas précédent dont nous tirons la conclusion :

la suite (z_{jm}) vérifie bien $a_j^k \leq z_{jm}$ pour k assez grand.

Il suffit ensuite de passer à la limite sur $n \to +\infty$ pour obtenir $\alpha_j^k \leq z_j$.

Nous avons ainsi montré que :

$$\max \{ x_j , \alpha_j^k \} \leq z_j \quad .$$

D'une manière similaire, il est possible de montrer que $z_j \leq \min \{ \beta_j^k , \bar{x}_j \}$

Nous pouvons en déduire que z est admissible pour $\overline{P}(x^k)$ pour k assez grand.
Mais dans ce cas $\overline{f}(z ; x^k) \geq \overline{f}(y^k ; x^k)$ pour k suffisamment grand, puisque y^k est optimum pour $\overline{P}(x^k)$.
Nous passons à la limite sur $k \to +\infty$:

$$\overline{f}(z ; x) \geq \overline{f}(y ; x)$$

et y est optimum pour $\overline{P}(x)$.

② Si $\exists\, i \in \{1, \dots, m\}$ tel que $\overline{f_i}(z ; x) - \hat{f_i} = 0$
l'hypothèse de Slater nous permet d'affirmer que : $\exists\, w$ tel que $\overline{f_i}(w ; x) - \hat{f_i} < 0$
Considérons $z_n = \left(1 - \frac{1}{n}\right) z + \frac{1}{n} w$

et $(z_n)_n \longrightarrow z$

Par le chapitre 2, $\overline{f_i}$ est convexe.

Donc

$$\overline{f_i}(z^m; x) \leq \left(1 - \frac{1}{m}\right) \overline{f_i}(z; x) + \frac{1}{m} \overline{f_i}(w; x)$$
$$< \left(1 - \frac{1}{m}\right) \hat{f_i} + \frac{1}{m} \hat{f_i}$$
$$\leq \hat{f_i}$$

Nous pouvons retourner au cas (1) car z^m vérifie $\overline{f_i}(z^m; x) - \hat{f_i} < 0 \quad i = 1 \dots m$

Nous en déduisons que z^m est admissible pour $\overline{P}(x^k)$ pour k assez grand et que

$$\overline{f^0}(z^m; x^k) \geq \overline{f^0}(y; x^k)$$

Nous passons à la limite sur $k \to +\infty$

et $m \to +\infty$

$$\overline{f^0}(z; x) \geq \overline{f^0}(y; x)$$

Et y est alors optimum pour $\overline{P}(x)$.

Il nous reste enfin à traiter <u>le cas où x^0 est non admissible pour P</u> :

Le domaine de $\overline{P}(x^0)$ est inclus dans X par le lemme 2. Si ce domaine est non vide, ce qui est le cas vu les hypothèses, $x^1, x^2, \dots$ seront admissibles pour $\overline{P}(x^0)$, $\overline{P}(x^1)$, ... et donc pour P.

Il suffit alors de considérer la suite (x^k) pour $k \geq 1$ et de se ramener ainsi au cas précédent.

C Q F D

§ 4.3 : Convergence avec recherche linéaire exacte.

La convergence de la méthode non modifiée nous a imposé des hypothèses assez fortes sur les fonctions, telle par exemple, celle de concavité de la fonction objectif et des fonctions des contraintes. Dans un souci d'alléger quelque peu ces hypothèses, nous avons introduit une recherche linéaire exacte et ceci par le biais d'une fonction de pénalisation exacte appelée $\phi_\lambda(x)$.

Ces modifications nous ont permis d'établir un second théorème.

Le problème P original est écrit sous une forme équivalente :

$$P \quad \begin{cases} \text{maximiser} & -f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f_i} \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \overline{x}_j \quad j = 1 \dots n \end{cases}$$

A un pb problème P, est associée la fonction de pénalisation exacte $\phi_\lambda(x)$ avec $\lambda > 0$ définie comme suit :

$$\phi_\lambda(x) = -f(x) - \lambda \sum_{i=1}^{m} \max \{ f_i(x) - \hat{f}_i, 0 \}$$

(nous allons considérer $\lambda > \Pi$ défini plus tard)

L'algorithme de résolution, quant à lui, contient un pas supplémentaire et devient :

Pas 0 : Initialisation : - choix du point de départ x^0
- poser $k = 0$

Pas 1 : Soit x^k.
Construction du sous-problème $\overline{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\overline{P}(x^k)$.
Soit z^k, sa solution optimale.

Pas 3 : Poser $x^{k+1} = \max_{[x^k, z^k]} \phi_\lambda(x)$

Pas 4 : $k \leftarrow k+1$
Retourner au pas 1.

La construction des sous-problèmes $\overline{P}(x^k)$ et des paramètres L, U, α, β suit le même procédé que celui employé lors de l'étude de la convergence sans recherche linéaire.

Ce procédé a été développé dans l'introduction de ce chapitre.

Le résultat de convergence que nous avons alors obtenu s'insère dans notre désir de compléter la méthode de K. Svanberg [4] par une étude personnelle de la convergence.
Ceci constitue donc, comme le théorème établi précédemment, l'originalité de la partie mathématique de ce travail.

Théorème 2 :

Sous les hypothèses suivantes :

- la fonction objectif f et les contraintes f_i ($i = 1 \dots m$) sont continûment différentiables.
- D est un polyèdre compact de $\mathbb{R}^n$ où $D = \{ t \in \mathbb{R}^n$ tels que $\underline{x}_j \leq t_j \leq \overline{x}_j \quad j = 1 \dots n \}$
- Le domaine de P est non vide.
- Une qualification de contraintes (Slater) est vérifiée en tout point du domaine de P.

Si (x^k) est une suite de points construite par l'algorithme précédent et si x^* est un point d'accumulation de cette suite,
alors x^* est un point de Kuhn-Tucker pour P.

La démonstration est, cette fois, directe. Elle se fait en deux étapes. [6]

Sous les hypothèses du théorème 2, il est possible de construire un autre problème P_1 à partir de P.

P_1 est de la forme :

$$P_1 \begin{cases} \text{maximiser} & -f(x) - q \sum_{j=1}^{m} z_j \\ \text{s.c.} & f_i(x) - z_i \leq \hat{f}_i \quad i = 1 \dots m \\ & (x, \{z_i \text{ tels que } i = 1 \dots m\}) \in D' \\ \text{où} & D' \equiv \{(x, \{z_i \; i = 1 \dots m\}) \text{ tels que} \\ & x \in D \quad \text{et } 0 \leq z_i \leq \Pi_1 \; i = 1 \dots m\} \\ \text{où} & \Pi_1 > \max_{1 \leq i \leq m} \{\max_{t \in D} |f_i(t) - \hat{f}_i|\} \end{cases}$$

Ce problème P_1 a l'avantage de vérifier quelques propriétés particulières.

Il nous est possible d'en étudier la convergence grâce au théorème suivant.

Remarquons que ce théorème est lui aussi, le fruit de notre étude personnelle.

Théorème 3 :

Sous les hypothèses suivantes :

- la fonction objectif et les contraintes f_i $(i = 1 \dots m)$ sont continûment différentiables.
- D est un polyèdre compact de $\mathbb{R}^n$ où $D = \{ t \in \mathbb{R}^n$ tels que $\underline{x}_j \leq t_j \leq \overline{x}_j \; j = 1 \dots n \}$
- le domaine de P est non vide.
- une qualification des contraintes (Slater) est vérifiée en tout point du domaine de P.
- $\forall x \in D$, le domaine de $\overline{P}(x)$ est non vide.
- les multiplicateurs de Lagrange associés à la solution optimale z^* de $\overline{P}(x)$ sont uniformément bornés par rapport à x : $\exists \Pi > 0$, $\forall x \in D$, z^* solution optimale de $\overline{P}(x)$, $v \in \mathbb{R}^{-m}$ multiplicateurs associés tels que $\max_{1 \leq i \leq m} |v_i| < \Pi$

Si (x^k) est une suite de points construite par l'algorithme et si x^* est un point d'accumulation de cette suite,

alors

x^* est un point de Kuhn-Tucker pour P

Il suffit alors, ensuite, de démontrer certaines relations d'équivalence entre les problèmes P et P_1, pour pouvoir déduire la convergence de P à partir de celle de P_1.

Dans une première partie, nous allons nous attacher à démontrer le théorème 3. Ensuite, il nous faudra montrer que le problème P_1 satisfait aux hypothèses de ce théorème. Et enfin, il restera à établir les relations d'équivalence.

4.3.1 : Démonstration du théorème 3.

Lemmes :

Lemme 5 : (Existence d'une dérivée directionnelle le long de d en x) :

$\forall x \in D$, $\forall d \in \mathbb{R}^n$
ϕ_λ possède une dérivée directionnelle le long de d en x égale à :

$$- \langle \nabla f(x), d \rangle - \lambda \sum_{i \in I^0(x)} \max \{ \langle \nabla f_i(x), d \rangle, 0 \}$$

$$- \lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle$$

où $I^0(x) = \{ i \in \{1, \dots, m\} \text{ tels que } f_i(x) - \hat{f}_i = 0 \}$

$I^+(x) = \{ i \in \{1, \dots, m\} \text{ tels que } f_i(x) - \hat{f}_i > 0 \}$

$I^-(x) = \{ i \in \{1, \dots, m\} \text{ tels que } f_i(x) - \hat{f}_i < 0 \}$

Démonstration du lemme 5 :

$\forall \theta$ suffisamment petit et $\theta > 0$, nous pouvons écrire le résultat suivant :

$$\frac{\phi_\lambda(x + \theta d) - \phi_\lambda(x)}{\theta} = \frac{-f(x + \theta d) + f(x)}{\theta}$$

$$- \lambda \sum_{i \in I^-(x)} \frac{\max\{f_i(x+\theta d) - \hat{f}_i, 0\} - \max\{f_i(x) - \hat{f}_i, 0\}}{\theta}$$

$$- \lambda \sum_{i \in I^+(x)} \frac{\max\{f_i(x+\theta d) - \hat{f}_i, 0\} - \max\{f_i(x) - \hat{f}_i, 0\}}{\theta}$$

$$- \lambda \sum_{i \in I^0(x)} \frac{\max\{f_i(x+\theta d) - \hat{f}_i, 0\} - \max\{f_i(x) - \hat{f}_i, 0\}}{\theta}$$

Considérons chacun des termes séparément et prenons la limite sur $\theta \to 0^+$.

Pour le premier terme :

$$\lim_{\theta \to 0^+} (-1) \cdot \frac{f(x + \theta d) - f(x)}{\theta} = - \langle \nabla f(x), d \rangle$$

puisque f est continûment différentiable.

Pour le second terme :

$i \in I^-(x)$ i.e. $f_i(x) - \hat{f}_i < 0$

i.e. $\max\{f_i(x) - \hat{f}_i, 0\} = 0$

Par continuité de f_i, pour θ suffisamment petit, nous aurons $f_i(x + \theta d) - \hat{f}_i \leq 0$

i.e. $\max\{f_i(x+\theta d) - \hat{f}_i, 0\} = 0$

Et donc

$$\lim_{\theta \to 0^+} \frac{\max\{f_i(x+\theta d) - \hat{f}_i, 0\} - \max\{f_i(x) - \hat{f}_i, 0\}}{\theta} = 0$$

et le second terme est alors nul.

Pour le troisième terme :

$i \in I^+(x)$ i.e. $f_i(x) - \hat{f}_i > 0$

i.e. $\max\{f_i(x) - \hat{f}_i, 0\} = f_i(x) - \hat{f}_i$

Par continuité de f_i, pour θ suffisamment petit, nous aurons $f_i(x + \theta d) - \hat{f}_i \geq 0$

i.e. $\max\{f_i(x+\theta d) - \hat{f}_i, 0\} = f_i(x+\theta d) - \hat{f}_i$

Et donc

$$\lim_{\theta \to 0^+} \frac{\max\{f_i(x+\theta d) - \hat{f}_i, 0\} - \max\{f_i(x) - \hat{f}_i, 0\}}{\theta}$$

$$= \lim_{\theta \to 0^+} \frac{f_i(x+\theta d) - \hat{f}_i - f_i(x) + \hat{f}_i}{\theta}$$

$$= \langle \nabla f_i(x), d \rangle$$

puisque f_i est continûment différentiable.

Et donc le troisième terme vaut :

$$-\lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle$$

Pour le dernier terme :

$$i \in I^0(x) \quad \text{i.e.} \quad f_i(x) - \hat{f}_i = 0$$

$$\text{i.e.} \quad \max \{ f_i(x) - \hat{f}_i, 0 \} = 0$$

Et donc

$$\lim_{\theta \to 0^+} \frac{\max \{ f_i(x+\theta d) - \hat{f}_i, 0 \} - \max \{ f_i(x) - \hat{f}_i, 0 \}}{\theta}$$

$$= \lim_{\theta \to 0^+} \frac{\max \{ f_i(x+\theta d) - \hat{f}_i, 0 \}}{\theta}$$

$$= \max \left\{ \lim_{\theta \to 0^+} \frac{f_i(x+\theta d) - \hat{f}_i}{\theta}, 0 \right\}$$

$$= \max \{ \langle \nabla f_i(x), d \rangle, 0 \}$$

puisque f_i est continûment différentiable.

Cela nous permet d'écrire :

$$\phi'_\lambda(x ; d) = - \langle \nabla f(x), d \rangle$$

$$- \lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle$$

$$- \lambda \sum_{i \in I^0(x)} \max \{ \langle \nabla f_i(x), d \rangle, 0 \}$$

CQFD

Lemme 6 :

$\forall x \in D$, $\forall d \in \mathbb{R}^n$ vérifiant :

$$f_i(x) - \hat{f}_i + \langle \nabla f_i(x), d \rangle \leq 0 \quad i = 1 \dots m$$

alors

$$\phi'_\lambda(x) = - \langle \nabla f(x), d \rangle - \lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle$$

où $I^+(x) = \{ i \in \{1, \dots, m\} \text{ tels que } f_i(x) - \hat{f}_i > 0 \}$

Démonstration du Lemme 6 :

Par le Lemme 5 , nous avons :

$$\phi'_\lambda(x ; d) = - \langle \nabla f(x), d \rangle - \lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle - \lambda \sum_{i \in I^0(x)} \max \{ \langle \nabla f_i(x), d \rangle , 0 \}$$

où $I^0(x) = \{ i \in \{1, \dots, m\} \text{ tels que } f_i(x) = \hat{f}_i \}$

Par hypothèse $\langle \nabla f_i(x), d \rangle \leq f_i(x) - \hat{f}_i$

et pour $i \in I^0(x)$ $\quad f_i(x) - \hat{f}_i = 0$

Donc, pour $i \in I^0(x)$, nous aurons :

$$\max \{ \langle \nabla f_i(x), d \rangle , 0 \} = 0$$

Et nous pouvons alors écrire :

$$\phi'_\lambda(x ; d) = - \langle \nabla f(x), d \rangle - \lambda \sum_{i \in I^+(x)} \langle \nabla f_i(x), d \rangle$$

CQFD

Lemme 7 :

Si x^* point de D n'est pas la solution optimale de $\overline{P}(x^*)$,
alors en notant $d = z^* - x^*$ où z^* est la solution optimale de $\overline{P}(x^*)$,
nous aurons $\phi'_\lambda(x ; d) > 0$
où $\lambda > \Pi$ et Π est la borne uniforme des multiplicateurs associés à z^*.

Démonstration du lemme 7 :

Nous allons considérer deux cas possibles :

① Si x^* n'appartient pas au domaine de P.
Comme, par hypothèse, $x^* \in D$, nous savons qu'alors

$$\exists i \in \{1, \dots, m\} \text{ tel que } f_i(x^*) - \hat{f}_i > 0$$

De plus, z^* est optimum pour $\overline{P}(x^*)$, z^* est donc admissible pour $\overline{P}(x^*)$ et nous pouvons écrire : $\overline{f_i}(z^* ; x^*) \leq \hat{f}_i \quad i = 1 \dots m$

Nous allons utiliser maintenant, plusieurs résultats démontrés au chapitre 2.
Et tout d'abord, la propriété sur la conservativité

des approximations .

Celle-ci s'énonçait comme suit :

Soient $\overline{f_i}(y;x)$ et $\overline{\overline{f_i}}(y;x)$
deux approximations construites avec les asymptotes $\{\underline{L}, \overline{U}\}$ et $\{\overline{\underline{L}}, \overline{\overline{U}}\}$.

Si $\underline{L}_j \leq \overline{\underline{L}}_j < x_j < \overline{\overline{U}}_j \leq \overline{U}_j \quad j = 1 \dots n$
alors $\forall y$ tel que $\overline{\underline{L}}_j < y_j < \overline{\overline{U}}_j$
nous aurons

$$\overline{f_i}(y;x) \leq \overline{\overline{f_i}}(y;x)$$

Nous avons aussi démontré que l'approximation construite sur les asymptotes $\{-\infty, +\infty\}$ était équivalente à la linéarisation directe.
Quelles que soient les asymptotes $\{L, U\}$ sur lesquelles est construite notre fonction $\overline{f_i}$, elles vérifieront nécessairement :

$$-\infty \leq L_j < U_j \leq +\infty \quad j = 1 \dots n$$

Donc, nous pouvons en déduire :

$$f_i(x^*) + \langle \nabla f_i(x^*), d \rangle \leq \overline{f_i}(z^*; x^*)$$

où $d = z^* - x^*$

Ce qui nous donne :

$$f_i(x^*) + \langle \nabla f_i(x^*), d \rangle \leq \hat{f}_i$$

Nous pouvons alors appliquer le lemme 6, qui

nous permet d'écrire :

$$\phi'_\lambda(x^*; d) = -\langle \nabla f(x^*), d\rangle - \lambda \sum_{i \in I_+(x^*)} \langle \nabla f_i(x^*), d\rangle$$

Nous savons que z^* est optimum pour $\overline{P}(x^*)$.
Si nous définissons le cône tangent à une partie fermée A en un de ses éléments x comme suit:

$$T(A, x) \equiv \{ y \in \mathbb{R}^n \text{ tels que}$$
$$\text{il existe } \{x^k \mid x^k \in A,\ k \in \mathbb{N}\},\ x^k \to x$$
$$\text{et } \{\lambda^k \mid k \in \mathbb{N},\ \lambda^k \in \mathbb{R}^+\}$$
$$\text{de sorte que } (\lambda^k (x^k - x) \text{ tq } k \in \mathbb{N}) \to y \}$$

et si nous entendons par cône polaire négatif d'une partie A de $\mathbb{R}^n$, l'ensemble

$$\Gamma^-(A) = \{ y \in \mathbb{R}^n \text{ tels que } \langle y, x\rangle \leq 0 \quad x \in A\}$$

nous pouvons alors écrire les conditions de Kuhn-Tucker de la manière suivante :

$$\begin{cases} \exists\ v \in \mathbb{R}^{-m} \text{ tel que} \\ -\nabla \overline{f}(z^*; x^*) + \sum_{i=1}^{m} v_i \nabla \overline{f}_i(z^*; x^*) \\ \qquad \text{appartient à } \Gamma^-(T(D^*, z^*)) \\ v_i (\overline{f}_i(z^*; x^*) - \hat{f}_i) = 0 \quad i = 1 \ldots m \end{cases}$$

où $D^* \equiv \{ t \in \mathbb{R}^n \text{ tels que } \max\{\underline{x}_j, \alpha_j^*\} \leq t_j$
$\text{et } t_j \leq \min\{\beta_j^*, \bar{x}_j\}\ j = 1..n\}$

Comme $x^* \in D$, nous avons $x^* \in D^*$ et $z^* \in D^*$.
Dans ce cas, $-z^* + x^*$ appartient à $T(D^*, z^*)$.

Ce qui nous permet d'écrire :

$$-\langle \nabla \overline{f}(z^*; x^*), d \rangle + \sum_{i=1}^{m} r_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle \geq 0$$

i.e. $$-\langle \nabla \overline{f}(z^*; x^*), d \rangle \geq -\sum_{i=1}^{m} r_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle$$

Par le chapitre 2, $\overline{f}$ est convexe :

$$\overline{f}(x^*; x^*) - \overline{f}(z^*; x^*) \geq -\langle \nabla \overline{f}(z^*; x^*), d \rangle$$

Donc nous avons :

$$\overline{f}(x^*; x^*) - \overline{f}(z^*; x^*) \geq -\sum_{i=1}^{m} r_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle$$

Mais comme $\overline{f}$ est convexe, nous avons aussi :

$$\overline{f}(z^*; x^*) - \overline{f}(x^*; x^*) \geq \langle \nabla \overline{f}(x^*; x^*), d \rangle$$

i.e. $$\overline{f}(x^*; x^*) - \overline{f}(z^*; x^*) \leq -\langle \nabla \overline{f}(x^*; x^*), d \rangle$$

Et donc

$$-\langle \nabla \overline{f}(x^*; x^*), d \rangle \geq -\sum_{i=1}^{m} r_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle$$

Par le chapitre 2, nous savons que :

$$\nabla \overline{f}(x^*; x^*) = \nabla f(x^*)$$

Nous avons alors une majoration de $\phi'_\lambda(x^*; d)$:

$$\phi'_\lambda(x^*; d) \geq -\sum_{i=1}^{m} r_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle - \lambda \sum_{i \in I^+(x^*)} \langle \nabla f_i(x^*), d \rangle$$

Analysons le premier terme du second membre :

* pour $i \in I_+(x^*)$:

Comme $\overline{f_i}$ est convexe :

$$\overline{f_i}(x^*; x^*) - \overline{f_i}(z^*; x^*) \geq \langle \nabla \overline{f_i}(z^*; x^*), -d \rangle$$

i.e. $\langle \nabla \overline{f_i}(z^*; x^*), d \rangle \geq \overline{f_i}(z^*; x^*) - \overline{f_i}(x^*; x^*)$

$$\geq \langle \nabla \overline{f_i}(x^*; x^*), d \rangle$$

Et comme $v_i \leq 0$:

nous avons :

$$-\sum_{i \in I_+(x^*)} v_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle \geq -\sum_{i \in I_+(x^*)} v_i \langle \nabla \overline{f_i}(x^*; x^*), d \rangle$$

$$= -\sum_{i \in I_+(x^*)} v_i \langle \nabla f_i(x^*), d \rangle$$

* pour $i \in I^0(x^*) \cup I^-(x^*)$:

comme $\overline{f_i}$ est convexe :

$$\overline{f_i}(x^*; x^*) - \overline{f_i}(z^*; x^*) \geq \langle \nabla \overline{f_i}(z^*; x^*), -d \rangle$$

i.e. $-v_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle \geq v_i \overline{f_i}(x^*; x^*) - v_i \overline{f_i}(z^*; x^*)$

Or z^* étant optimal, nous avons :

$$v_i (\overline{f_i}(z^*; x^*) - \hat{f_i}) = 0$$

i.e. $v_i \overline{f_i}(z^*; x^*) = v_i \hat{f_i}$

Ce qui nous donne :

$$-v_i \langle \nabla \overline{f_i}(z^*; x^*), d \rangle \geq v_i \overline{f_i}(x^*; x^*) - v_i \hat{f_i}$$

$$\geq v_i (\overline{f_i}(x^*; x^*) - \hat{f_i})$$

$$\geq 0$$

Donc pour $i \in I^0(x^*) \cup I^-(x^*)$:

$$-v_i < \nabla \overline{f_i}(z^*; x^*), d > \geq 0$$

Reprenons alors $\phi'_\lambda(x^*; d)$ et nous obtenons :

$$\phi'_\lambda(x^*; d) \geq -\sum_{i \in I^+(x^*)} v_i < \nabla f_i(x^*), d >$$

$$- \lambda \sum_{i \in I^+(x^*)} < \nabla f_i(x^*), d >$$

$$\geq -\sum_{i \in I^+(x^*)} (v_i + \lambda) < \nabla f_i(x^*), d >$$

Or nous avions établi que :

$$f_i(x^*) + < \nabla f_i(x^*), d > \leq \hat{f_i}$$

Donc pour $i \in I^+(x^*)$:

$$0 < f_i(x^*) - \hat{f_i} \leq - < \nabla f_i(x^*), d >$$

i.e. $$0 < - < \nabla f_i(x^*), d >$$

Donc $\phi'_\lambda(x^*; d)$ sera strictement positif pour autant que nous aurons $v_i + \lambda > 0$.
Ce qui est le cas si les multiplicateurs v_i sont uniformément bornés par Π, et si nous choisissons $\lambda > \Pi$.

② si x^* appartient au domaine de P :
Dans ce cas, $\forall\, i \in I^+(x^*)$

et $\phi'_\lambda(x^*; d) = -\langle \nabla f(x^*), d \rangle > 0$

En effet, en utilisant le même raisonnement que dans le cas précédent, nous obtenons :

$$\phi'_\lambda(x^*; d) = -\langle \nabla f(x^*), d \rangle + 0$$

De plus, $-\langle \nabla f(x^*), d \rangle$ doit être strictement positif sinon ? x^* serait solution optimale de ? $\overline{P}(x^*)$, ce qui est impossible vu les hypothèses.

CQFD

Lemme 8 :

Soit la fonction multivoque ψ définie $\forall x \in \mathbb{R}^n$ par

$$\psi(x) = \{ x \text{ admissibles pour } \overline{P}(x) \}$$

alors ψ est une application ouverte et fermée en tout x^* admissible pour P. [7]

Démonstration du Lemme 8 :

① Montrons d'abord que c'est une application fermée : Soit $(x^k) \longrightarrow x^*$

$(y^k) \longrightarrow y^*$

où $\forall k$, y^k est admissible pour $\overline{P}(x^k)$

y^* est-il alors admissible pour $\overline{P}(x^*)$?

En effet, vérifions que $\overline{f_i}(y^*; x^*) \leq \hat{f}_i \quad i = 1 \dots m$

Nous connaissons l'expression de $\overline{f_i}(y^*; x^*)$:

$$\overline{f_i}(y^*; x^*) = f(x^*) +$$

$$\sum_{j \in J+} \left[\frac{(v_j^* - x_j^*)^2}{v_j^* - y_j^*} - (v_j^* - x_j^*) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^*}$$

$$+ \sum_{j \in J-} \left[(x_j^* - z_j^*) - \frac{(x_j^* - z_j^*)^2}{y_j^* - z_j^*} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^*}$$

où $\begin{cases} J+ = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^*} \geq 0 \} \\ J- = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^*} < 0 \} \end{cases}$

Par hypothèse, y^k est admissible pour $\overline{P}(x^k)$.

Donc $\overline{f_i}(y^k; x^k) \leq \hat{f}_i$

ou bien

$$f_i(x^k) + \sum_{j \in J'+} \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j^k} - (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

$$+ \sum_{j \in J'-} \left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j^k - z_j^k} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \leq \hat{f}_i$$

où $\begin{cases} J'+ = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} \geq 0 \} \\ J'- = \{ j \in \{1, \dots, n\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} < 0 \} \end{cases}$

Il suffit alors de passer à la limite pour $k \to +\infty$.
Par le lemme 4, les sommes sur $J+$ et $J'+$ se feront sur les mêmes indices (ainsi que pour $J-$ et $J'-$), à partir d'un certain rang.
De plus, $x^k \longrightarrow x^*$

$y^k \longrightarrow y^*$

Et, par définition, nous avons

$$\lim_{k \to +\infty} u_j^k = \lim_{k \to +\infty} x_j^k + (\bar{x}_j - \underline{x}_j) = u_j^*$$

$$\lim_{k \to +\infty} l_j^k = \lim_{k \to +\infty} x_j^k + (\bar{x}_j - \underline{x}_j) = l_j^*$$

Nous obtenons alors $\overline{f}_i(y^*; x^*) \leq \hat{f}_i \quad i = 1 \dots m$

Nous vérifions ensuite que

$$\max\{\underline{x}_j, \alpha_j^*\} \leq y_j^* \leq \min\{\beta_j^*, \bar{x}_j\} \quad j = 1 \dots n$$

Comme y^k est admissible pour $\overline{P}(x^k)$:

$$\max\{\underline{x}_j, \alpha_j^k\} \leq y_j^k \leq \min\{\beta_j^k, \bar{x}_j\} \quad j = 1 \dots n$$

De plus

$$\lim_{k \to +\infty} \alpha_j^k = \alpha_j^* \quad \text{et} \quad \lim_{k \to +\infty} \beta_j^k = \beta_j^*$$

Nous avons donc bien y^* admissible pour $\overline{P}(x^*)$

② Montrons maintenant que c'est une application

ouverte : soit $t \in \phi(x^*)$

y-a-t-il une suite $(t^k) \longrightarrow t$ telle que

$\exists k_0$ tel que $\forall k > k_0$

t^k soit admissible pour $\overline{P}(x^k)$?

Il nous est possible de montrer que t est admissible pour $\overline{P}(x^k)$ pour k suffisamment grand.

Nous pouvons alors considérer la suite

$(t^k) = (t)$ suite constante.

Pour démontrer que t est admissible pour $\overline{P}(x^k)$ pour k assez grand, nous allons traiter deux cas :

a) si $\forall i = 1 \dots m \quad \overline{g_i}(t ; x^*) - \hat{g}_i = -\varepsilon_i < 0$

Dans ce cas, t est admissible pour $\overline{P}(x^k)$.

En effet,

$\exists k_0$ tel que $\forall k \geq k_0 \quad \overline{g_i}(t ; x^k) \leq \hat{g}_i$

Sinon, nous aurions

$\forall k \quad \exists n_k \geq k$ tel que $\overline{g_i}(t ; x^{n_k}) > \hat{g}_i$

Considérons la sous-suite $\left(\overline{g_i}(t ; x^{n_k}) - \hat{g}_i \right)_k$

Nous aurons :

$$\left(\overline{g_i}(t ; x^{n_k}) - \hat{g}_i \right)_k \longrightarrow \left(\overline{g_i}(t ; x^*) - \hat{g}_i \right)$$

et donc $\left(\overline{f_i}\ (t ; x^{nk}) - \widehat{f_i} \right) \longrightarrow 0 - \varepsilon_i$

Ce qui est absurde puisque la sous-suite est choisie de telle sorte que

$$\forall k \quad \overline{f_i}\ (t ; x^{nk}) - \widehat{f_i} > 0$$

Il nous faut encore montrer que t vérifie :

$$\max \{ \underline{x}_j , \alpha_j^k \} \leq t_j \leq \min \{ \beta_j^k , \bar{x}_j \} \quad j = 1 \ldots n$$

Montrons tout d'abord la première égalité :

$$\max \{ \underline{x}_j , \alpha_j^k \} \leq t_j$$

Deux cas se présentent :

- si $\max \{ \underline{x}_j , \alpha_j^k \} = \underline{x}_j \qquad j \in J^1$

  Dans ce cas, comme t est admissible pour $\overline{P}(x^*)$ $\quad t_j \geq \underline{x}_j$

- si $\max \{ \underline{x}_j , \alpha_j^k \} = \alpha_j^k \qquad j \in J^2$

  alors $\alpha_j^k \leq t_j$?

  * si $\forall j \in J^2$, nous avons $\alpha_j^* < t_j$

    alors pour k suffisamment grand, nous aurons $\alpha_j^k \leq t_j$

    sinon $\forall k \ \exists\, nk$ tel que $\alpha_j^{nk} > t_j$

    Or $\left(\alpha_j^{nk} \right)_k \longrightarrow \alpha_j^* \geq t_j$

    Comme nous avons supposé $\alpha_j^* < t_j$, cette hypothèse est absurde et nous obtenons donc bien $\alpha_j^k \leq t_j$

* Si $\exists j \in J^2$ tel que $a_j^* = t_j$

Par hypothèse, a_j^* est choisi tel que $a_j^* < x_j^*$.

Soit $t_{jn} = \left(1 - \frac{1}{n}\right) t_j + \frac{1}{n} x_j$

Nous observons que $(t_{jn})_n \longrightarrow t_j$

et $a_j^* - t_{jn} = a_j^* - \left(1 - \frac{1}{n}\right) t_j - \frac{1}{n} x_j$

$= a_j^* - t_j + \frac{1}{n}(t_j - x_j)$

$= a_j^* - t_j + \frac{1}{n}(a_j^* - x_j) - \frac{1}{n}(a_j^* - t_j)$

$= (a_j^* - t_j)\left(1 - \frac{1}{n}\right) + \frac{1}{n}(a_j^* - x_j)$

< 0

Et donc $a_j^* < t_{jn} \qquad \forall j \in J^2$

Nous nous retrouvons avec t_{jn} dans le cas précédent dont nous tirons la conclusion : la suite (t_{jn}) vérifie $a_j^k \leq t_{jn}$ pour k assez grand.

Il suffit ensuite de passer à la limite sur $n \longrightarrow +\infty$ pour obtenir $a_j^k \leq t_j$.

Nous avons ainsi montré que $\max\{x_j, a_j^k\} \leq t_j$.

D'une manière similaire, il est possible de montrer que $t_j \leq \min\{\bar{x}_j, \beta_j^k\}$.

Nous pouvons en déduire que t est admissible pour $\bar{P}(x^k)$ pour k assez grand.

b) Si $\exists i \in \{1, \dots, m\}$ tel que $\overline{f_i}(t; x^*) - \hat{f}_i = 0$

l'hypothèse de Slater nous permet d'affirmer que $\exists w$ tel que $\overline{f_i}(w; x^*) - \hat{f}_i < 0$

Considérons $t_n = \left(1 - \frac{1}{n}\right) t + \frac{1}{n} w$

et $(t_n)_n \longrightarrow t$

Par le chapitre 2, $\overline{f_i}$ est convexe :

$$\overline{f_i}(t_n; x^*) \leq \left(1 - \frac{1}{n}\right) \overline{f_i}(t; x^*) + \frac{1}{n} \overline{f_i}(w; x^*)$$
$$< \left(1 - \frac{1}{n}\right) \hat{f}_i + \frac{1}{n} \hat{f}_i$$
$$\leq \hat{f}_i$$

Nous pouvons retourner au cas a) car t_n vérifie

$$\overline{f_i}(t_n; x^*) - \hat{f}_i < 0 \qquad i = 1 \dots m$$

Nous en déduisons que t_n est admissible pour $\overline{P}(x^k)$ pour k assez grand.

En passant à la limite sur $n \longrightarrow +\infty$, nous obtenons aisément que t est admissible pour $\overline{P}(x^k)$ pour k assez grand.

CQFD

En nous servant de ces divers résultats, il nous est possible de démontrer le théorème 3.

Théorème 3 :

Sous les hypothèses suivantes :

- la fonction objectif f et les contraintes f_i $(i = 1 \dots m)$ sont continûment différentiables.
- D est un polyèdre compact de $\mathbb{R}^n$.
  où $D = \{ t \in \mathbb{R}^n$ tels que $\underline{x}_j \leq t_j \leq \bar{x}_j \quad j = 1 \dots n \}$
- le domaine de P est non vide.
- une qualification des contraintes (Slater) est vérifiée en tout point du domaine de P.
- $\forall x \in D$, le domaine de $\bar{P}(x)$ est non vide.
- les multiplicateurs de Lagrange associés à la solution optimale z^* de $\bar{P}(x)$ sont uniformément bornés par rapport à x :
  $\exists \Pi > 0$, $\forall x \in D$, z^* solution optimale de $\bar{P}(x)$, $u \in \mathbb{R}^m$ multiplicateurs associés tels que $\max_{1 \leq i \leq m} |u_i| < \Pi$

Si (x^k) est une suite de points construite par l'algorithme et si x^* est un point d'accumulation de cette suite,

alors

x^* est un point de Kuhn-Tucker pour P.

Le problème P dont il s'agit est du type :

$$P \quad \begin{cases} \text{maximiser} & -f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots n \end{cases}$$

Démonstration :

Soit z^k la solution optimale de $\overline{P}(x^k)$.

Considérons deux cas :

① Supposons $\exists\, k_0$ tel que $x^{k_0+1} = x^{k_0}$

Donc les suites $(x^k)_{\mathbb{N}}$ et $(z^k)_{\mathbb{N}}$ sont constantes à partir du rang k_0.

Dans ce cas $\begin{cases} \lim\limits_{k \to +\infty} z^k = z^* = z^{k_0} \\ \lim\limits_{k \to +\infty} x^k = x^* = x^{k_0} \end{cases}$

Et, par définition, x^* maximise $\phi_\lambda(x)$ sur $[x^*, z^*]$, donc

$$\phi'_\lambda(x^*; d^*) \leq 0 \quad \text{où} \quad d = z^* - x^*$$

Par le lemme 7, x^* est optimum pour $\overline{P}(x^*)$.

Par le lemme 2, x^* est point de Kuhn-Tucker pour P.

(e) Supposons que $\forall k \quad x^{k+1} \neq x^k$.

Donc, les suites $(x^k)_{\mathbb{N}}$ et $(z^k)_{\mathbb{N}}$ sont des suites infinies.

or (z^k) est une suite de points dans un compact. Elle admet alors un point d'accumulation.

Nous avons alors : $\exists N' \subset \mathbb{N}$ tel que

$$\left.\begin{array}{ll} (x^k)_{k \in N'} & \longrightarrow x^* \\ (x^{k+1})_{k \in N'} & \longrightarrow x^{**} \\ (z^k)_{k \in N'} & \longrightarrow z^* \end{array}\right\} \; ? \quad a_0 \in D ?$$

Soit $t \in [x^*, z^*]$:

$\exists \mu \in [0,1]$ tel que $t = x^* + \mu d^*$

où $d = z^* - x^*$

Soit la suite

$(t^k$ tel que $t^k = x^k + \mu d^k = x^k + \mu (z^k - x^k)$ où $k \in \mathbb{N})$.

Cette suite converge vers un point t.

or, par construction, x^{k+1} maximise $\phi_\lambda(x)$ sur $[x^k, z^k]$ et donc

$$\phi_\lambda(t^k) \leq \phi_\lambda(x^{k+1})$$

Nous passons à la limite sur $k \in N' \subset \mathbb{N}$.

Nous avons par continuité de $\phi_\lambda(x)$:

$$\phi_\lambda(t) \leq \phi_\lambda(x^{**})$$

Comme D est compact, $\phi_\lambda(x)$ est bornée supérieurement sur D.

Par construction, $(\phi_\lambda(x^k)$ où $k \in \mathbb{N})$ est

$a_1 \in D ?$

strictement croissante.
Elle est donc convergente :

$$\left(\phi_\lambda (x^k)\right)_k \longrightarrow \phi_\lambda^*$$

Donc, toute sous-suite converge vers la même limite : $\phi_\lambda^* = \phi_\lambda (x^*) = \phi_\lambda (x^{**})$.
Et donc, x^* maximise ϕ_λ sur $[x^*, z^*]$.

Considérons la suite $(z^k$ tel que $k \in N' \subset \mathbb{N}) \longrightarrow z^*$.
Nous avons vu, dans le lemme 8, que l'application multivoque qui, à tout point du domaine de P, associe l'ensemble des solutions optimales de $\bar{P}(x)$, est une application ouverte et fermée.
Si, donc, nous réussissons à montrer que x^* est admissible pour P, z^* sera optimum pour $\bar{P}(x^*)$.
Dans ce cas, supposons par l'absurde que x^* ne soit pas point de Kuhn-Tucker pour P, alors par le lemme 2, x^* n'est pas solution optimale de $\bar{P}(x^*)$.
Par le lemme 7, nous affirmons alors

$$\phi'_\lambda (x^*; d) > 0 \quad \text{où} \quad d = z^* - x^*.$$

Mais ceci contredit le fait que x^* maximise $\phi_\lambda(x)$ sur le segment $[x^*, z^*]$.
Donc, x^* ne peut être que point de Kuhn-Tucker

pour P.

Il nous reste donc à montrer que x^* est bien admissible pour P.

Démontrons cela par l'absurde :

Comme $x^* \in D$, si x^* n'est pas admissible pour P, nous pouvons écrire :

$$\exists\, i = i_0 \text{ tel que } f_{i_0}(x^*) - \hat{f}_{i_0} > 0$$

$$\text{i.e. } \overline{f}_{i_0}(x^*; x^*) - \hat{f}_{i_0} > 0$$

puisque nous avons vu, au chapitre 2, que :

$$f_{i_0}(x^*) = \overline{f}_{i_0}(x^*; x^*)$$

Alors x^* n'est pas admissible pour $\overline{P}(x^*)$.

Et, dans ce cas, x^k n'est pas admissible pour $\overline{P}(x^k)$ pour k suffisamment grand.

En effet, sinon nous aurions :

$$\overline{f}_{i_0}(x^k; x^k) - \hat{f}_{i_0} \leq 0$$

$$\text{i.e } f_{i_0}(x^k) - \hat{f}_{i_0} \leq 0$$

et en passant à la limite sur $k \to +\infty$:

$$f_{i_0}(x^*) - \hat{f}_{i_0} \leq 0$$

Ce qui est en contradiction avec le fait que nous ayons $f_{i_0}(x^*) - \hat{f}_{i_0} > 0$.

x^k n'étant pas admissible pour $\overline{P}(x^k)$, x^k ne sera pas solution optimale pour $\overline{P}(x^k)$.

Posons $d^k \equiv z^k - x^k$ où z^k est, par définition, solution optimale de $\overline{P}(x^k)$.

Par le lemme 7, nous aurons :

$$\phi'_\lambda(x^k ; d^k) = - \langle \nabla f(x^k), d^k \rangle - \lambda \sum_{i \in I^+(x^k)} \langle \nabla f_i(x^k), d^k \rangle > 0$$

avec $f_i(x^k) - \hat{f}_i \leq - \langle \nabla f_i(x^k), d^k \rangle$

En passant à la limite sur $k \to 0 + \infty$ dans cette dernière inégalité, nous aurons :

$$f_i(x^*) - \hat{f}_i \leq - \langle \nabla f_i(x^*), d \rangle$$

Les Hypothèses du lemme 6 sont alors vérifiées.

Nous en déduisons :

$$\phi'_\lambda(x^* ; d) = - \langle \nabla f(x^*), d \rangle - \lambda \sum_{i \in I^+(x^*)} \langle \nabla f_i(x^*), d \rangle$$

Comparons les ensembles $I^+(x^k)$ et $I^+(x^*)$.

Nous pouvons établir que $I^+(x^k) = I^+(x^*) \cup I$

où $I = \{ i \in \{1, \dots, m\}$ tels que

$$\forall k \quad f_i(x^k) - \hat{f}_i > 0$$

$$\text{et } \lim_{k \to 0 + \infty} x^k = x^*$$

$$\text{et } f_i(x^*) - \hat{f}_i = 0 \quad \}$$

En effet, si $i_0 \in I^+(x^k)$ $\quad f_{i_0}(x^k) - \hat{f}_{i_0} > 0$

En passant à la limite sur $k \to 0 + \infty$:

$$f_{i_0}(x^*) - \hat{f}_{i_0} \geq 0$$

Et si $i_0 \in I^+(x^*) \cup I$:

$\begin{cases} \text{si } i_0 \in I \text{ , par construction de } I \text{ , } i_0 \in I^+(x^k) \\ \text{si } i_0 \in I^+(x^*) \text{ , à partir d'un certain rang, nous aurons nécessairement } i_0 \in I^+(x^k) \end{cases}$

Considérons l'expression :

$$\phi'_\lambda(x^k; d^k) = - \langle \nabla f(x^k), d^k \rangle - \lambda \sum_{i \in I^+(x^k)} \langle \nabla f_i(x^k), d^k \rangle$$

$$= - \langle \nabla f(x^k), d^k \rangle - \lambda \sum_{i \in I^+(x^*)} \langle \nabla f_i(x^k), d^k \rangle - \lambda \sum_{i \in I} \langle \nabla f_i(x^k), d^k \rangle$$

Rappelons-nous que z^k est solution optimale pour $\overline{P}(x^k)$. Nous avons alors comme développé dans le lemme 7 :

$$\phi'_\lambda(x^k; d^k) \geq - \sum_{i=1}^{m} v_i \langle \nabla \overline{f_i}(z^k; x^k), d^k \rangle - \lambda \sum_{i \in I^+(x^k)} \langle \nabla f_i(x^k), d^k \rangle$$

$$\geq - \sum_{i \in I^+(x^k)} v_i \langle \nabla \overline{f_i}(z^k; x^k), d^k \rangle - \lambda \sum_{i \in I^+(x^k)} \langle \nabla f_i(x^k), d^k \rangle$$

$$= - \sum_{i \in I^+(x^*)} v_i \langle \nabla \overline{f_i}(z^k; x^k), d^k \rangle$$

$$- \sum_{i \in I} v_i < \nabla \overline{f_i}\,(z^k ; x^k), d^k > - \lambda \sum_{i \in I^+(x^k)} < \nabla f_i(x^k), d^k >$$

$$\geq - \sum_{i \in I^+(x^*)} v_i < \nabla f_i(x^k), d^k > + \sum_{i \in I} v_i \,(f_i(x^k) - \hat{f}_i) - \lambda \sum_{i \in I^+(x^k)} < \nabla f_i(x^k), d^k >$$

Passons à la limite sur $k \longrightarrow +\infty$:

$$\phi'_\lambda(x^*; d) - \lambda \sum_{i \in I} < \nabla f_i(x^*), d >$$

$$\geq - \sum_{i \in I^+(x^*)} v_i < \nabla f_i(x^*), d > + \sum_{i \in I} v_i \,(f_i(x^*) - \hat{f}_i) - \lambda \sum_{i \in I^+(x^*)} < \nabla f_i(x^*), d > - \lambda \sum_{i \in I} < \nabla f_i(x^*), d >$$

Or pour $i \in I$ $\quad f_i(x^*) - \hat{f}_i = 0$

Donc

$$\phi'_\lambda(x^*; d) \geq - \sum_{i \in I^+(x^*)} v_i < \nabla f_i(x^*), d > - \lambda \sum_{i \in I^+(x^*)} < \nabla f_i(x^*), d >$$

$$= - \sum_{i \in I^+(x^*)} (v_i + \lambda) < \nabla f_i(x^*), d >$$

Or par hypothèse $\quad v_i + \lambda > 0$

Et $\quad 0 < f_i(x^*) - \hat{f}_i \leq - < \nabla f_i(x^*), d > \quad i \in I^+(x^*)$

Donc $\phi'_\lambda (x^*; d) > 0$

Mais ceci contredit le fait que x^* maximise ϕ_λ sur $[x^*, z^*]$ où $d = x^* - z^*$.
Cela nous prouve que x^* est admissible pour P.

Ayant montré dans le lemme 8 que l'application qui, à tout point x du domaine de P, associe l'ensemble des points admissibles du problème $\overline{P}(x)$, est une application ouverte et fermée sur le domaine de P, il suffit d'appliquer le théorème du maximum de [8] pour obtenir que z^* soit solution optimale de $\overline{P}(x^*)$.
Voici ce théorème :

Soient $P \subset \mathbb{R}^n$, $Q \subset \mathbb{R}^p$, $q \in Q$, $T \subset \mathbb{R}^+$

$A : Q \longrightarrow \mathcal{P}(P)$

$f : P \times Q \longrightarrow \mathbb{R}$

On pose $\phi(q) = \sup \{ f(p,q) \mid p \in A(q), q \text{ fixé} \}$

$\Pi(q, \varepsilon) = \{ p \in A(q) \mid f(p,q) \geq \phi(q) - \varepsilon \}$

On suppose $\begin{cases} A(q) \neq \phi \\ \Pi(q, \varepsilon) \neq \phi \end{cases}$

et ce $\forall (q, \varepsilon) \in Q \times T$

Sous ces conditions, nous avons :

(i) P compact
A fermée en $\dot{q}$
ϕ fermée sur $A(\dot{q}) \times \{\dot{q}\}$

$\Rightarrow$ ϕ est fermée en $\dot{q}$

(ii) A ouverte en $\dot{q}$
ϕ ouverte sur $A(\dot{q}) \times \{\dot{q}\}$

$\Rightarrow$ ϕ est ouverte en $\dot{q}$

(iii) A fermée et ouverte en $\dot{q}$
ϕ continue sur $A(\dot{q}) \times \{\dot{q}\}$

$\Rightarrow$ Π est fermée sur $\{\dot{q}\} \times T$

Il suffit alors d'utiliser le théorème en considérant :

$A \equiv \phi$

$Q \equiv$ ensemble des points admissibles pour P

$\dot{q} \equiv x^*$

$P \equiv D$

$\phi(p, q) = -\overline{p}(p; q)$

4.3.2 : Le problème P_η vérifie les hypothèses du théorème 3.

Rappelons - nous la façon dont nous avions construit le problème P_η :

$$P_1 \begin{cases} \text{maximiser} & -f(x) - q \sum_{i=1}^{m} z_i \\ \text{s.c.} & f_i(x) - z_i \leq \hat{f}_i \quad i = 1 \dots m \\ & (x, \{z_i \quad i = 1 \dots m\}) \in D' \\ \text{où} & D' = \{(x, \{z_i \quad i = 1 \dots m\}) \text{ tels que} \\ & x \in D \quad 0 \leq z_i \leq \eta_1 \quad i = 1 \dots m\} \\ \text{où} & \eta_1 > \max_{1 \leq i \leq m} \{\max_{t \in D} |f_i(t) - \hat{f}_i|\} \end{cases}$$

Rappelons - nous également que le théorème 3 s'applique à un problème du type :

$$\begin{cases} \text{maximiser} & -f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & x \in D \end{cases}$$

Dans une première étape, montrons qu'il est possible d'exprimer P_1 sous cette forme :

$$\begin{cases} \text{maximiser} & -g(z) \\ \text{s.c.} & g_i(z) \leq \hat{f}_i \quad i = 1 \dots m \\ & z \in D' \end{cases}$$

où nous avons posé :

$$g(z) = -f(x) - q \cdot \sum_{i=1}^{m} \lambda_i$$

$$z = (x, \{\lambda_i \quad i = 1 \dots m\})$$

$$g_i(z) = f_i(x) - \lambda_i$$

Il nous reste donc à vérifier que P_1 satisfait aux hypothèses du théorème 3. Cela sera fait dans les lemmes qui suivent.

Il est cependant nécessaire de se rappeler le but de cette étude qui est la convergence de l'algorithme appliqué au problème original P.

Pour cela, nous avons signalé le résultat que nous avons obtenu dans le théorème 2.

Pour démontrer ce théorème, nous avons construit à partir de P, un autre problème P_1. Nous possédons, donc, un problème Original satisfaisant les hypothèses du théorème 2 et voulons construire à partir de celui-là, un problème P_1 répondant aux conditions du théorème 3.

Voyons si c'est le cas.

Lemme 9 : (P_1 satisfait la première Hypothèse) :

La fonction objectif et les contraintes de P_1 sont continûment différentiables.

Démonstration du Lemme 9 :

La fonction objectif est $-f(x) - q \sum_{i=1}^{m} z_i$.
f étant, par Hypothèse, continûment différentiable sur $\mathbb{R}^n$, le résultat est évident.
Les contraintes sont $f_i(x) - z_i$.
f_i étant, par Hypothèse, continûment différentiables sur $\mathbb{R}^n$, le résultat est immédiat.

C Q F D

Lemme 10 : (P_1 satisfait la deuxième Hypothèse) :

D' est un polyèdre compact de $\mathbb{R}^{n+m}$

Démonstration du Lemme 10 :

D' est défini comme suit :

$D' = \{ (x, \{ z_i \; i = 1 \dots m \}) \text{ tels que}$

$x \in D \qquad 0 \leq z_i \leq \Pi_1 \qquad i = 1 \dots m \}$

où $\eta_1 > \max_{1 \le i \le m} \{ \max_{t \in D} |f_i(t) - \hat{f}_i| \}$

D est, par hypothèse, un polyèdre compact.
De plus, les fonctions f_i sont continues sur $\mathbb{R}^n$.
Elles sont donc bornées supérieurement sur D.
Ce qui nous permet d'affirmer que η_1 existe et est fini.
Mais, dans ce cas, D' sera compact sur $\mathbb{R}^{n+m}$

CQFD

Lemme 11 : (P_1 satisfait la troisième hypothèse):

Le domaine de P_1 est non vide.

Démonstration du Lemme 11:

Remarquons que η_1 est toujours, par définition, positif.
De plus, par hypothèse, le domaine de P est non vide. Nous pouvons donc dire qu'il existe x appartenant à D et tel que $f_i(x) - \hat{f}_i \le 0 \quad i = 1 \dots m$
Considérons alors

$(x, \{\eta_i \; i = 1 \dots m\})$ où $\eta_i = 0 \quad i = 1 \dots m$

Ce point appartient bien à D' et nous vérifions:

$f_i(x) - 0 \leq \hat{f}_i \quad i = 1 \dots m$

Ce point appartient donc au domaine de P_2 qui n'est pas vide.

CQFD

Lemme 2 : (P_2 satisfait la 4e Hypothèse) :

Une qualification des contraintes (Slater) est vérifiée en tout point du domaine de P_2.

Démonstration du Lemme 2 :

Montrons d'abord que P_2 vérifie la condition de Slater.

Nous savons que P vérifie cette condition, donc

$\exists\, x \in D$ tel que $f_i(x) - \hat{f}_i < 0 \quad i = 1 \dots m$

Comme η_i est positif par définition, nous aurons quel que soit η_i :

$$f_i(x) - \hat{f}_i < \eta_i \qquad i = 1 \dots m$$

Il est possible de choisir η_i non nul puisque η_2 est non nul.

En effet, si $\eta_2 = 0$ nous aurions :

$$0 > \max_{1 \leq i \leq m} \left\{ \max_{x \in D} | f_i(x) - \hat{f}_i | \right\}$$
$$\geq \max_{1 \leq i \leq m} \left\{ | f_i(x) - \hat{f}_i | \right\} > 0$$

Donc Π_1 est non nul et l'hypothèse de Slater est vérifiée.

Considérons maintenant le problème :

$$\overline{P_2}\ (\{x, \{r_i\ i=1\dots m\}\})$$

Vérifie-t-il la condition de Slater ?

Voyons d'abord comment ce problème se formule :

$$\begin{cases} \text{maximiser } -\overline{\varphi}(y;x) - q\sum_{i=1}^{m}\left[(r_i - L'_i) - \dfrac{(r_i - L'_i)^2}{p_i - L'_i}\right] \\ \text{s.c. } \overline{\varphi_i}(y;x) - \left[(r_i - L'_i) - \dfrac{(r_i - L'_i)^2}{p_i - L'_i}\right] \le \hat{\varphi}_i \quad i=1\dots m \\ (y, \{p_i\ i=1\dots m\}) \in D'' \\ \text{où } D'' \text{ est tel que :} \\ \max\{\underline{x}_j, \alpha_j\} \le y_j \le \min\{\beta_j, \overline{x}_j\} \quad j=1\dots n \\ \max\{0, \alpha'_i\} \le p_i \le \min\{\beta'_i, \Pi'_1\} \quad i=1\dots m \\ \text{où } \Pi'_1 > \Pi_1 \end{cases}$$

où L'_i $i=1\dots m$ sont les asymptotes associées aux variables $\{r_i\}$ et α'_i, β'_i $i=1\dots m$ sont les "move-limits" qui y sont associées.

Par hypothèse, nous savons qu'il existe un point z tel que :

$\overline{f_i}(z;x) < \hat{f}_i$

et $\max\{\underline{x}_j, a_j\} < z_j < \min\{\beta_j, \bar{x}_j\}$

Donc, si nous considérons le point suivant :

$(z, \{\eta_i \quad i = 1 \dots m\})$

nous pouvons écrire :

$$\overline{f_i}(z;x) - \left[(\eta_i - L'_i) - \frac{(\eta_i - L'_i)^2}{\eta_i - L'_i}\right]$$

$$= \overline{f_i}(z;x) < \hat{f}_i$$

Par définition, nous avons $\alpha'_i < \eta_i < \beta'_i$.

Mais ne pourrait-il que $\eta_i = 0$?

Dans ce cas, comme nous avons vu que η_1 est non nul, nous sommes certains qu'il existe un point appartenant à $]0, \eta_1[$. Il suffit alors de considérer ce point au lieu de $\eta_i = 0$.

En effet, appelons ce point η'_i.

Comme $\eta'_i > \eta_i$ alors

$$\frac{(\eta_i - L'_i)^2}{\eta'_i - L'_i} < \frac{(\eta_i - L'_i)^2}{\eta_i - L_i}$$

et donc $\overline{f_i}(z;x) - \left[(\eta_i - L'_i) - \frac{(\eta_i - L'_i)^2}{\eta'_i - L'_i}\right]$

$$\leq \overline{f_i}(z;x) - \left[(\eta_i - L'_i) - \frac{(\eta_i - L'_i)^2}{\eta_i - L'_i}\right] < \hat{f}_i$$

Nous pouvons donc bien considérer un tel point. Comme nous avons substitué Π_1 par Π_1', il est impossible d'avoir $\pi_i = \Pi'_1$. Cette opération est valable vu la façon dont Π_1 a été défini. La condition de Slater est donc bien vérifiée.

CQFD

Lemme 13: (P_1 satisfait la cinquième hypothèse):

$\forall (x, \{\pi_i \ i = 1 \dots m\})$,
le domaine de $\overline{P}((x, \{\pi_i \ i = 1 \dots m\}))$
est non vide.

Démonstration du Lemme 13:

La démonstration est immédiate puisque ce problème satisfait la condition de Slater comme cela a été démontré dans le Lemme 12.

CQFD

Lemme 14: (P_1 satisfait la sixième hypothèse):

Les multiplicateurs de Lagrange associés à la solution optimale $(z^*, \{\pi_i^* \ i = 1 \dots m\})$ de

$\overline{P}(x, \{\eta_i \; i=1\dots m\})$ sont uniformément bornés.

$\exists \Pi > 0$, $\forall (y, \{p_i\}) \in D''$,

$(z^*, \{\eta_i^*\})$ solution optimale de $\overline{P}((x, \{\eta_i\}))$

$v_i \in \mathbb{R}^-$, multiplicateurs associés tels que

$$\max_{1 \le i \le m} |v_i| < \Pi .$$

Démonstration du lemme 24 :

Il nous faut écrire les conditions de Kuhn-Tucker en $(z^*, \{\eta_i^*, i=1\dots m\})$:

$$\begin{cases} \exists\, v_i \in \mathbb{R}^- \quad i=1\dots m \quad \text{tels que} \\ -\nabla \overline{f}(z^*; x) + \sum_{i=1}^{m} v_i \nabla \overline{f_i}(z^*; x) \in \Gamma^-(T(D^*, z^*)) \\ -q \dfrac{(\eta_i - L'_i)^2}{(\eta_i^* - L'_i)^2} - v_i \dfrac{(\eta_i - L'_i)^2}{(\eta_i^* - L'_i)^2} \le 0 \quad i=1\dots m \\ \left(\overline{f}_i(z^*; x) - \left[(\eta_i - L'_i) - \dfrac{(\eta_i - L'_i)^2}{\eta_i^* - L'_i} \right] - \hat{f}_i \right) v_i = 0 \quad i=1\dots m \end{cases}$$

où $\begin{cases} L'_i \text{ sont les asymptotes associées à } \eta_i . \\ D^* \equiv \{ t \text{ tels que } \quad j=1\dots n \\ \qquad \max\{\underline{x}_j, \alpha_j\} \le t_j \le \min\{\overline{x}_j, \beta_j\} \} \\ \Gamma^-(T(D^*, z^*)) \text{ a été défini précédemment.} \end{cases}$

Nous en tirons : $(-q - v_i) \dfrac{(\eta_i - L'_i)^2}{(\eta_i^* - L'_i)^2} \le 0$

$$\text{i.e.} \quad -q - v_i \leq 0$$
$$\text{i.e.} \quad -v_i \leq q$$
$$\text{i.e.} \quad |v_i| \leq q \qquad i = 1 \ldots m$$

Il suffit alors de choisir $\Pi \geq q$

CQFD

Nous avons ainsi réussi à montrer que le problème P_1 satisfait aux hypothèses du théorème 3.

4.3.3 : Relations d'équivalence entre les problèmes P_1 et P.

Partant d'un problème P vérifiant les conditions du théorème 2, nous avons construit un autre problème P_1 auquel nous pouvons appliquer le théorème 3 comme cela a été démontré en 4.3.2.

Nous pouvons en conclure :

Si $((x^k, \{z_i^k \; i = 1 \ldots m\}))$ est une suite de points construite par l'algorithme,

Si $(x^*, \{z_i^* \; i = 1 \ldots m\})$ est un point d'accumulation de cette suite,

alors
$(x^*, \{\eta_i^*\})$ est un point de Kuhn-Tucker pour P_2.

Si nous parvenons à montrer que, dans ce cas, x^* est point de Kuhn-Tucker pour P, nous aurons démontré ainsi le théorème 2.

Dans ce but, voici deux lemmes qui permettent d'établir des relations d'équivalence entre P et P_2.

Lemme 15 :

Si $(x^*, \{\eta_i^* \; i = 1 \dots m\})$ est un point de Kuhn-Tucker pour P_2
et si $\eta_i^* = 0$ pour $i = 1 \dots m$
alors
x^* est point de Kuhn-Tucker pour P.

Démonstration du Lemme 15 :

Montrons tout d'abord, que x^* est bien admissible pour P.
Or par définition, $x^* \in D$,
et $f_i(x^*) - \hat{f_i} \leq \eta_i^* = 0 \qquad i = 1 \dots m$

Donc x^* est admissible pour P.

Montrons ensuite que x^* vérifie les conditions d'optimalité pour P.
Comme $(x^*, \{\eta_i^*\})$ satisfait les conditions de Kuhn-Tucker pour P_η, nous avons:

$$\begin{cases} -\nabla f(x^*) - \sum_{i=1}^{m} u_i^* \nabla f_i(x^*) \in \Gamma^-(T(D, x^*)) \\ -q - u_i^* \leq 0 \quad i = 1 \dots m \\ (f_i(x^*) - \eta_i^* - \hat{f}_i) u_i^* = 0 \quad i = 1 \dots m \end{cases}$$

En remplaçant $\eta_i^* = 0$, nous obtenons:

$$\begin{cases} -\nabla f(x^*) - \sum_{i=1}^{m} u_i^* \nabla f_i(x^*) \in \Gamma^-(T(D, x^*)) \\ (f_i(x^*) - \hat{f}_i) u_i^* = 0 \quad i = 1 \dots m \end{cases}$$

Nous retrouvons alors les conditions de Kuhn-Tucker en x^* pour P.

CQFD

Lemme 16 :

Si $\forall q > 0$, $(x^*, \{\eta_i^*\})$ est optimum pour P_η alors nous avons $\eta_i^* = 0 \quad i = 1 \dots m$

Démonstration du Lemme 16 :

Nous allons démontrer cela par l'absurde.

Supposons qu'il existe $\pi_{i_0}^* \neq 0$.

Considérons

$$q_0 > \frac{\max_{t \in D} (-f(t)) - \min_{t \in D} (-f(t))}{\pi_{i_0}^*} > 0$$

Comme le domaine de P est non vide par hypothèse, soit z y appartenant.

Dans ce cas, le point $(z, \{r_i = 0 \; i = 1 \dots m\})$ est admissible pour P_z de façon immédiate.

Ecrivons :

$$-f(z) + f(x^*) + q_0 \pi_{i_0}^*$$
$$> -f(z) + f(x^*) + \max_{t \in D} (-f(t)) - \min_{t \in D} (-f(t))$$

par définition de q_0.

Donc

$$-f(z) + f(x^*) + q_0 \pi_{i_0}^*$$
$$> \underbrace{-f(z) - \min_{t \in D} (-f(t))}_{\geq 0} + \underbrace{\max_{t \in D} (-f(t)) + f(x^*)}_{\geq 0}$$
$$\geq 0$$

Donc $-f(z) + f(x^*) + q_0 \pi_{i_0}^* > 0$

$$i.e. \quad -f(z) > -f(x^*) - q_0 \eta_{i_0}^* > -f(x^*) - q_0 \eta_{i_0}^* - q_0 \sum_{\substack{i=1 \\ i \neq i_0}}^{m} \eta_i^*$$

car $q_0 > 0$ et $\eta_i^* \geq 0$ $\quad i = 1 \dots m$

Mais alors le point $(z, \{\eta_i = 0 \quad i = 1 \dots m\})$ serait meilleure solution que $(x^*, \{\eta_i^* \quad i = 1 \dots m\})$. Ce qui est impossible puisque $(x^*, \{\eta_i^*\})$ est solution optimale.
Donc l'hypothèse $\eta_{i_0}^* = 0$ est absurde.

C Q F D

4.3.4 : Conclusion.

En conclusion, reprenons le schéma général de la démonstration du théorème 2.

Théorème 2 :

Sous les hypothèses suivantes :

- la fonction objectif f et les contraintes f_i $i = 1 \dots m$ sont continûment différentiables
- D est un polyèdre compact de $\mathbb{R}^n$

× le domaine de P est non vide

× une qualification des contraintes (Slater) est vérifiée en tout point du domaine de P.

Si (x^k) est une suite de points construite par l'algorithme développé précédemment et si x^* est un point d'accumulation de cette suite,

alors x^* est un point de Kuhn-Tucker pour P.

Démonstration du théorème 2 :

Construisons à partir du problème P

$$P \quad \begin{cases} \text{maximiser} & -f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \qquad i = 1 \dots m \\ & x \in D \end{cases}$$

le problème P_1 comme suit :

$$P_1 \quad \begin{cases} \text{maximiser} & -f(x) - q \sum_{i=1}^{m} r_i \\ \text{s.c.} & f_i(x) - r_i \leq \hat{f}_i \quad i = 1 \dots m \\ & (x, \{ r_i \; i = 1 \dots m \}) \in D' \end{cases}$$

où $D' \equiv \{ (x, \{ r_i \; i = 1 \dots m \}) \text{ tels que}$

$$x \in D \qquad 0 \leq r_i \leq \Pi_2 \qquad i = 1 \dots m \}$$

où $\Pi_1 > \max\limits_{1 \leq i \leq m} \{ \max\limits_{t \in D} | f_i(t) - \hat{f}_i | \}$

et où $q > 0$.

Nous avons montré que ce problème P_2 vérifie les hypothèses du théorème 3. Ce théorème ayant été démontré, nous pouvons l'appliquer au problème P_2.

Nous obtenons alors $(x^*, \{ r_i^* \})$ point de Kuhn-Tucker pour P_1.

Le lemme 16 nous indique alors que

$$r_i^* = 0 \qquad i = 1 \dots m .$$

Il suffit alors de conclure par le lemme 15 qui nous affirme que x^* est point de Kuhn-Tucker pour P.

C Q F D

Chapitre 5 :

"Irréalisabilité" du point de départ.

§5.1 : Analyse :

Nous avons détaillé dans le chapitre 3, le principe suivi pour construire les sous-problèmes $\overline{P}(x^k)$ à chaque itération.
Et nous avions obtenu la forme suivante :

$$\overline{P}(x^k) \begin{cases} \text{minimiser} & \overline{f}(y ; x^k) \\ \text{s.c.} & \overline{f_i}(y ; x^k) \leq \hat{f}_i \quad i = 1 \dots m \\ & \max\{\underline{x}_j, \alpha_j^k\} \leq y_j \leq \min\{\beta_j^k, \bar{x}_j\} \\ & \qquad j = 1 \dots n \end{cases}$$

Même si nous supposons que l'intervalle bornant y n'est pas vide, il se pourrait, et plus particulièrement dans les premières itérations, qu'un mauvais choix du point de départ rende l'un ou l'autre des sous-problèmes irréalisable (i.e. sans solution réalisable).

Ceci peut être très ennuyeux.
En effet, si nous nous rappelons le processus itératif à la base de la méthode :

Pas 0 : Initialisation : - choix du point de départ x^0
- poser $k = 0$

Pas 1 : Soit x^k.

Construction du sous-problème $\overline{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\overline{P}(x^k)$.

Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$

Retourner au pas 1.

nous remarquons qu'il est indispensable de savoir résoudre à chaque itération le sous-problème pour pouvoir passer à l'itération suivante.

D'un point de vue pratique, il nous faudrait donc pouvoir éviter ce problème pour ne pas bloquer le processus.

Nous pourrions nous restreindre à considérer des points de départ "réalisables", ce qui éliminerait nos problèmes.

Cependant, il serait plus intéressant de modifier quelque peu la méthode en vue de pouvoir tenir en compte n'importe quel point de départ.

Il serait aussi d'un grand intérêt d'essayer de se rapprocher le plus possible d'une solution réalisable et donc de calculer une solution

qui soit le moins irréalisable possible.

Pour résoudre ces problèmes, nous allons introduire dans chaque sous-problème, des variables dites "artificielles".

§ 5.2 : Introduction de variables artificielles.

Pour des raisons développées précédemment, nous avons introduit des variables artificielles dans l'expression d'un sous-problème $\overline{P}(x^k)$. Il s'exprime alors :

$$\widetilde{P}(x^k) \begin{cases} \text{minimiser } \overline{f}(y; x^k) + \sum_{i=1}^{m} (d_i z_i + d_i z_i^2) \\ \text{s.c. } \overline{f}_i(y; x^k) - z_i \leq \hat{f}_i \quad i = 1 \dots m \\ \max\{\underline{x}_j, \alpha_j^k\} \leq y_j \leq \min\{\overline{x}_j, \beta_j^k\} \quad j = 1 \dots m \\ z_i \geq 0 \quad i = 1 \dots m \end{cases}$$

Nous noterons un tel sous-problème $\widetilde{P}(x^k)$.

Les modifications consistent en l'ajout d'une fonction quadratique à la fonction objectif et de fonctions linéaires aux contraintes.

Ces fonctions font intervenir des nouvelles variables z_i $\quad i = 1 \ldots m$.

Remarquons que nous avons ajouté une variable par contrainte.

Les contraintes de bornes relatives aux z_i sont des contraintes de positivité.

Les d_i, $i = 1 \ldots m$, sont des paramètres dont les valeurs sont à fixer.

Un sous-problème de ce type admet toujours une solution réalisable si l'intervalle bornant les variables n'est pas vide. En effet, quel que soit x, il est toujours possible de trouver z pour satisfaire les contraintes.

Le choix des valeurs des paramètres d_i n'est pas fixé. Il faut cependant les choisir comme étant des nombres réels "relativement grands". Il suffit de les prendre très grands vis-à-vis des autres coefficients de la fonction objectif.

§ 5.3 : Equivalence en cas de réalisabilité.

Nous avons donc pu éviter le problème de

la non-réalisabilité d'un sous-problème $\overline{P}(x^k)$ en résolvant dans ce cas, le problème $\widetilde{P}(x^k)$.

Mais il est assez contraignant de devoir vérifier si le sous-problème peut admettre une solution réalisable ou non, afin de savoir si nous devons résoudre $\overline{P}(x^k)$ ou $\widetilde{P}(x^k)$.

Il nous vient alors l'idée de résoudre $\widetilde{P}(x^k)$ automatiquement. Mais il nous faut vérifier que si, par hasard, le problème $\overline{P}(x^k)$ était réalisable, en résolvant à sa place $\widetilde{P}(x^k)$, nous obtenons bien la même solution.

Il nous a été possible d'établir, à ce propos, le résultat suivant (cfr. Annexe 5.A.):

Il y a équivalence entre les problèmes $\overline{P}(x)$ et $\widetilde{P}(x)$ lorsque $\overline{P}(x)$ est réalisable.

i.e. si $\overline{P}(x)$ est réalisable

$(x^*, z^*=0)$ est solution optimale de $\widetilde{P}(x)$

$\Rightarrow$ x^* est solution optimale de $\overline{P}(x)$.

Dans ce cas, la suite d'itérées que nous obtenons par l'algorithme initial ou par l'algorithme modifié que voici:

Pas 0 : Initialisation : - choix du point de départ x^0
- poser $k = 0$

Pas 1 : Soit x^k.
Construction du sous-problème $\tilde{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\tilde{P}(x^k)$.
Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$.
Retourner au pas 1.

est la même, si le problème de non-réalisabilité ne se pose pas.

Si, par contre, un tel problème se pose, en résolvant $\tilde{P}(x^k)$, nous obtenons un ou plusieurs z_i strictement positifs à l'optimum. Mais vu le coût élevé de ces variables (coefficients di grands), elles ont une valeur minimale et en ce sens, la solution x correspondante sera aussi réalisable que possible.

D'une manière générale, nous préférerons donc en pratique, considérer l'algorithme modifié en résolvant $\tilde{P}(x^k)$ au lieu de $\bar{P}(x^k)$ à chaque itération.

Annexe au chapitre 5.

Annexe 5.A. : Equivalence de $\overline{P}(x)$ et $\widetilde{P}(x)$.

Si $\overline{P}(x)$ est réalisable :

$(x^*, z^* = 0)$ est solution optimale de $\widetilde{P}(x)$

$\Rightarrow$ x^* est solution optimale de $\overline{P}(x)$

Démontrons la première implication $\Rightarrow$:

x^* est admissible pour $\overline{P}(x)$, donc $(x^*, 0)$ est admissible pour $\widetilde{P}(x)$.

La fonction objectif de $\widetilde{P}(x)$ en $(x^*, 0)$ vaut :

$$\overline{f}(x^*; x) + \sum_{i=1}^{m} (d_i \cdot 0 + d_i \cdot 0^2)$$

$$= \overline{f}(x^*; x)$$

Supposons, par l'absurde, qu'il existe (w^*, w^*) solution optimale de $\widetilde{P}(x)$ et que $(x^*, 0)$ ne le soit pas.

Alors nous aurons que la fonction objectif de $\widetilde{P}(x)$ en (w^*, w^*) sera strictement inférieure à celle en $(x^*, 0)$:

$$\overline{f}(w^*; x) + \sum_{i=1}^{m} (d_i w_i^* + d_i w_i^{*2}) < \overline{f}(x^*; x)$$

$$\overline{f}(w^*; x) - \overline{f}(x^*; x) + \sum_{i=1}^{m} (d_i w_i^* + d_i w_i^{*2}) < 0$$

Or $\overline{f}(w^*; x) - \overline{f}(x^*; x) \geq 0$ car x^* est solution optimale de $\overline{P}(x)$.

Et $\sum_{i=1}^{m} (d_i w_i^* + d_i w_i^{*2}) \geq 0$ si $d_i \geq 0 \ \forall i$

ce qui est le cas si d_i est suffisamment grand.

Donc

$$\overline{\varphi}(v^*; x) - \overline{\varphi}(x^*; x) + \sum_{i=1}^{m} (d_i w_i^* + d_i w_i^{*2}) \geq 0$$

ce qui apporte une contradiction.

Démontrons la réciproque $= 0$:

C'est évident car $(x^*, z^* = 0)$ solution optimale de $\tilde{P}(x)$ implique que $(x^*, z^* = 0)$ est encore solution optimale si nous ne considérons que les problèmes $\tilde{P}(x)$ avec les variables z_i fixées à 0. x^* est alors solution optimale de $\overline{P}(x)$.

Chapitre 6 :

Etude du duel du sous-problème linéarisé.

Pour résoudre un sous-problème $\tilde{P}(x^k)$, nous proposons une méthode duale vu les propriétés du sous-problème qui, d'après les remarques faites dans les chapitres précédents, est convexe et séparable par construction des approximations.

C'est pourquoi, dans ce chapitre, nous allons analyser le duel d'un sous-problème $\tilde{P}(x^k)$.

§ 6.1 : Formulation.

Formulons, tout d'abord, le duel de façon classique.

Rappelons-nous l'expression de $\tilde{P}(x^k)$:

$$\tilde{P}(x^k) \begin{cases} \text{minimiser} & \overline{\phi}(y; x^k) + \sum_{i=1}^{m} (d_i z_i + d_i z_i^2) \\ \text{s.c.} & \overline{\phi_i}(y; x^k) - z_i \leq \hat{\phi}_i \quad i = 1 \dots m \\ & \max\{\underline{x}_j, \alpha_j^k\} \leq y_j \leq \min\{\bar{x}_j, \beta_j^k\} \\ & \qquad\qquad\qquad\qquad j = 1 \dots n \\ & z_i \geq 0 \quad i = 1 \dots m \end{cases}$$

Le duel en sera :

$$D(x^k) \begin{cases} \text{maximiser} & \mathcal{L}(\lambda) \\ \text{s.c.} & \lambda \geq 0 \end{cases}$$

$$\text{où } \mathcal{L}(\lambda) = \min_{\substack{y_j \in D_j \quad j=1\dots n \\ z_i \geq 0 \quad i=1\dots m}} \left[\overline{f}(y; x^k) + \sum_{i=1}^{m} (d_i z_i + d_i z_i^2) \right.$$

$$\left. + \sum_{i=1}^{m} \lambda_i \left(\overline{f_i}(y; x^k) - z_i - \hat{f}_i \right) \right]$$

$$\text{où } D_j \equiv \left[\max \{ \underline{x}_j, a_j^k \}, \min \{ \beta_j^k, \bar{x}_j \} \right]$$

Je nous est possible d'exploiter la séparabilité et d'arriver à remplacer le minimum "général", en ce sens que nous y minimisons sur toutes les variables, en une somme de minima n'opérant chacun d'entre eux que sur une variable.

Après calculs consistant à remplacer les approximations $\overline{f}$ et $\overline{f_i}$ par leur expression tirée du chapitre 2, nous arrivons à ceci (cfr Annexe 6.A.) :

$$\mathcal{L}(\lambda) = f(x^k) + \sum_{i=1}^{m} \lambda_i \left(f_i(x^k) - \hat{f}_i \right)$$

$$+ \sum_{i=1}^{m} \min_{z_i \geq 0} \left(d_i z_i + d_i z_i^2 - \lambda_i z_i \right)$$

$$+ \sum_{j=1}^{n} \min_{y_j \in D_j} Z_j(y_j ; \lambda)$$

où $\quad \Gamma_j(y_j ; \lambda) =$

si $\dfrac{\partial f}{\partial y_j}\Big|_{x^k} \geq 0$

$$\left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k)\right]\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ \sum_{i \in I_j^-} \lambda_i \left[(x_j^k - \zeta_j^k) - \frac{(x_j^k - \zeta_j^k)^2}{y_j - \zeta_j^k}\right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

si $\dfrac{\partial f}{\partial y_j}\Big|_{x^k} < 0$

$$\left[(x_j^k - \zeta_j^k) - \frac{(x_j^k - \zeta_j^k)^2}{y_j - \zeta_j^k}\right]\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ \sum_{i \in I_j^+} \lambda_i \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k)\right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

où $\begin{cases} I_j^+ = \{ i \in \{1, \dots, m\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} \geq 0 \} \\ I_j^- = \{ i \in \{1, \dots, m\} \text{ tels que } \dfrac{\partial f_i}{\partial y_j}\Big|_{x^k} < 0 \} \end{cases}$

§ 6.2 : Exploitation de la séparabilité.

A partir de l'expression précédente du problème dual, comme nous avons réussi à transformer la fonction duale de façon à avoir une somme de minima ne faisant intervenir qu'une variable, nous allons pouvoir en déduire l'expression des variables du sous-problème primal en fonction des variables duales.

6.2.1 : Expression des variables artificielles.

Reprenons la fonction duale telle que nous l'avons établie ci-dessus :

$$\mathcal{L}(\lambda) = f(x^k) + \sum_{i=1}^{m} \lambda_i (f_i(x^k) - \hat{f}_i)$$

$$+ \sum_{i=1}^{m} \min_{z_i \geq 0} (d_i z_i + d_i z_i^2 - \lambda_i z_i)$$

$$+ \sum_{j=1}^{n} \min_{y_j \in D_j} \Gamma_j(y_j ; \lambda)$$

Nous remarquons que seule, la partie suivante comporte la variable z_i :

$$\min_{z_i \geq 0} (d_i z_i + d_i z_i^2 - \lambda_i z_i)$$

Nous pouvons donc, en calculant ce minimum, exprimer la variable artificielle z_i en fonction de la variable duale λ_i :

Calcul de la dérivée première : $d_i - \lambda_i + 2 d_i z_i$

Annulation de cette dérivée : $d_i - \lambda_i + 2 d_i z_i = 0$

$$\text{i.e.} \quad z_i = \frac{\lambda_i}{2 d_i} - \frac{1}{2}$$

Calcul de la dérivée seconde : $2 d_i > 0$

si d_i est suffisamment grand.

C'est donc bien un minimum.

Nous aurons alors :

$$z_i = \begin{cases} 0 & \text{si } \frac{\lambda_i}{2 d_i} - \frac{1}{2} \leq 0 \\ \frac{\lambda_i}{2 d_i} - \frac{1}{2} & \text{sinon} \end{cases}$$

6.2.2 : Expression des variables primales.

Par un procédé semblable, il nous est possible d'établir une relation entre chacune des variables primales y_j $j = 1 \dots n$ et les variables duales λ_i $i = 1 \dots m$.

En effet, revoici encore la fonction duale :

$$\mathcal{L}(\lambda) = f(x^k) + \sum_{i=1}^{m} \lambda_i \left(f_i(x^k) - \hat{f}_i \right)$$

$$+ \sum_{i=1}^{m} \min_{z_i \geq 0} \left(d_i z_i + d_i z_i^2 - \lambda_i z_i \right)$$

$$+ \sum_{j=1}^{m} \min_{y_j \in D_j} Z_j(y_j ; \lambda)$$

La variable y_j n'apparaît que dans le terme suivant :

$$\min_{y_j \in D_j} Z_j(y_j ; \lambda)$$

où $D_j = \left[\max \{ \underline{x}_j, a_j^k \} , \min \{ \beta_j^k, \bar{x}_j \} \right]$

Similairement à ce qui a été fait pour les variables artificielles, nous pouvons calculer ce minimum.

Voici, tout d'abord, la dérivée première :

$$\frac{\partial Z_j}{\partial y_j} = - \frac{(v_j^k - x_j^k)^2}{(v_j^k - y_j)^2} \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right]$$

$$+ \sum_{i \in I_j^-} \lambda_i \frac{(x_j^k - \ell_j^k)^2}{(y_j - \ell_j^k)^2} \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \quad \text{si } \frac{\partial f}{\partial y_j}\Big|_{x^k} \geq 0$$

$$\frac{(x_j^k - \ell_j^k)^2}{(y_j - \ell_j^k)^2} \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right]$$

$$+ \sum_{i \in I_j^+} \lambda_i \frac{(v_j^k - x_j^k)^2}{(v_j^k - \mathcal{U}_j)^2} \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \quad \text{si } \frac{\partial f}{\partial y_j}\Big|_{x^k} < 0$$

Divisons le travail en 2 parties :

Etudions en premier lieu le cas où $\frac{\partial f}{\partial y_j}|_{x^k} \geq 0$

Analysons diverses possibilités suivant les informations que nous pouvons tirer de la dérivée seconde :

$$\frac{\partial^2 L_j}{\partial y_j^2} = \frac{(v_j^k - x_j^k)^2}{(v_j^k - y_j)^3} \left[\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right]$$

$$- \sum_{i \in I_j^-} \lambda_i \frac{(x_j^k - \zeta_j^k)^2}{(y_j - \zeta_j^k)^3} \frac{\partial f_i}{\partial y_j}|_{x^k}$$

Et nous avons $\frac{\partial^2 L_j}{\partial y_j^2} \geq 0$ par définition des différents termes.

Et $\frac{\partial^2 L_j}{\partial y_j^2}$ ne sera nul que si

$$\left[\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right] = 0$$

et $$\left[- \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right] = 0$$

Etudions ce cas séparément.

Nous voyons alors que $L_j(y_j ; \lambda) = 0$

et donc n'importe quelle valeur comprise dans

$$[\max\{\underline{x}_j, \alpha_j^k\}, \min\{\beta_j^k, \bar{x}_j\}]$$

pour y_j, minimise $\Gamma_j(y_j;\lambda)$.

Si nous mettons de côté ce cas particulier, nous arrivons logiquement à la conclusion que

$$\frac{\partial^2 \Gamma_j}{\partial y_j^2} > 0$$

et donc que $\Gamma_j(y_j;\lambda)$ est strictement convexe.

Si, alors, $\left.\frac{\partial \Gamma_j}{\partial y_j}\right|_{\max\{\bar{x}_j, \alpha_j^k\}} \geq 0$

comme $\Gamma_j(y_j;\lambda)$ est strictement convexe, $\frac{\partial \Gamma_j}{\partial y_j}$ sera strictement croissante et n'admettra donc pas de racine sur notre intervalle.

Le minimum de $\Gamma_j(y_j;\lambda)$ sur ce même intervalle sera alors atteint en

$$y_j = \max\{\bar{x}_j, \alpha_j^k\}$$

Un raisonnement parallèle nous permet de dire que

si $\left.\frac{\partial \Gamma_j}{\partial y_j}\right|_{\min\{\beta_j^k, \bar{x}_j\}} \leq 0$

alors le minimum de $\Gamma_j(y_j;\lambda)$ sera atteint en $y_j = \min\{\beta_j^k, \bar{x}_j\}$.

Je mais reste alors à traiter le cas où

$$\frac{\partial L_j}{\partial y_j}\Big|_{\max\{\underline{x}_j,\ a_j^k\}} < 0$$

et $$\frac{\partial L_j}{\partial y_j}\Big|_{\min\{\beta_j^k,\ \bar{x}_j\}} > 0$$

Comme $\frac{\partial L_j}{\partial y_j}$ est strictement croissante, il y a une racine sur l'intervalle

$$\left[\max\{\underline{x}_j, a_j^k\},\ \min\{\beta_j^k, \bar{x}_j\}\right].$$

Calculons-la.

Il mais faut pour cela, annuler $\frac{\partial L_j}{\partial y_j}$ et en calculer les racines.

Après de longs calculs détaillés en Annexe 6.B., mais obtenons les racines suivantes:

$$\left[(v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} z_j^k\right.$$

$$\left(v_j^k - x_j^k\right)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} \mp (x_j^k - z_j^k)\left(-\sum_{i\in I_j^-}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}$$

$$+\ v_j^k\,(x_j^k - z_j^k)\left(-\sum_{i\in I_j^-}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}$$

$$\left.\left(-(x_j^k - z_j^k)\left(-\sum_{i\in I_j^-}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} \pm (v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}\right)\right]$$

$$\Big/\left[(v_j^k - x_j^k)\,e\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right) + (x_j^k - z_j^k)\left(\sum_{i\in I_j^-}\lambda_i\frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)\right]$$

Remarquons qu'elles ne sont définies que pour autant que

$$\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right) \quad \text{et} \quad \left(\sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)$$

ne soient pas nuls simultanément.
Nous avons déjà traité ce cas précédemment.
Toutefois, il est aisé de voir que l'une des deux racines n'appartient pas à l'intervalle considéré (cfr. Annexe 6.C.).
Nous obtenons alors la valeur de y_j qui minimise $L_j(y_j; \lambda)$:

$$y_j = \left[\zeta_j^k (U_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{7/2} + U_j^k (x_j^k - \zeta_j^k) \left(-\sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{7/2} \right] /$$

$$\left[(U_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{7/2} + (x_j^k - \zeta_j^k) \left(-\sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{7/2} \right]$$

Etudions le cas où $\frac{\partial f}{\partial y_j}\Big|_{x^k} < 0$:

Le développement est tout à fait semblable à celui effectué dans le cas où $\frac{\partial f}{\partial y_j}\Big|_{x^k} \geq 0$.

Nous donnons, de suite, les résultats auxquels nous aboutissons :

si $\left.\frac{\partial L_j}{\partial y_j}\right|_{\max\{\underline{x}_j, a_j^k\}} \geq 0$ $\quad y_j = \max\{\underline{x}_j, a_j^k\}$

si $\left.\frac{\partial L_j}{\partial y_j}\right|_{\min\{\beta_j^k, \bar{x}_j\}} \leq 0$ $\quad y_j = \min\{\beta_j^k, \bar{x}_j\}$

si $\left.\frac{\partial L_j}{\partial y_j}\right|_{\max\{\underline{x}_j, a_j^k\}} < 0$ et $\left.\frac{\partial L_j}{\partial y_j}\right|_{\min\{\beta_j^k, \bar{x}_j\}} > 0$

soit y_j est libre

soit

$$y_j = \left[v_j^k (x_j^k - l_j^k) \left(-\frac{\partial f}{\partial y_j}\Big|_{x^k} - \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} + l_j^k (v_j^k - x_j^k) \left(\sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right] \Big/ \left[(x_j^k - l_j^k) \left(-\frac{\partial f}{\partial y_j}\Big|_{x^k} - \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} + (v_j^k - x_j^k) \left(\sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right]$$

En conclusion, voici l'expression de la variable y_j en fonction des variables duales :

- si $\left.\frac{\partial L_j}{\partial y_j}\right|_{\max\{\underline{x}_j, a_j^k\}} \geq 0$ $\quad y_j = \max\{\underline{x}_j, a_j^k\}$

- si $\left.\frac{\partial L_j}{\partial y_j}\right|_{\min\{\beta_j{}^k, \bar{x}_j\}} \leq 0$ $\quad y_j = \min\{\beta_j{}^k, \bar{x}_j\}$

- sinon

si $\left.\frac{\partial f}{\partial y_j}\right|_{x^k} \geq 0$

| si $\left(\left.\frac{\partial f}{\partial y_j}\right|_{x^k} + \sum_{i \in I_j^+} \lambda_i \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}\right) = 0$
| et $\left(-\sum_{i \in I_j^-} \lambda_i \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}\right) = 0$
| alors y_j est libre
| sinon y_j a la valeur détaillée p. 6.10

si $\left.\frac{\partial f}{\partial y_j}\right|_{x^k} < 0$

| si $\left(\left.\frac{\partial f}{\partial y_j}\right|_{x^k} + \sum_{i \in I_j^-} \lambda_i \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}\right) = 0$
| et $\left(\sum_{i \in I_j^+} \lambda_i \left.\frac{\partial f_i}{\partial y_j}\right|_{x^k}\right) = 0$
| alors y_j est libre
| sinon y_j a la valeur détaillée p. 6.22

§ 6.3 : Expression du gradient de la fonction duale.

Comme il est courant en programmation non linéaire, l'expression des différentes composantes du gradient de la fonction duale, est donnée par les contraintes primales évaluées en les variables primales calculées d'après les variables duales où nous estimons le gradient.

Nous obtenons alors :

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \overline{f}_i(y; x^k) - z_i - \hat{f}_i$$

où $y = y(\lambda)$ $\quad z_i = z_i(\lambda)$

§ 6.4 : Expression du Hessien de la fonction duale.

Il est aussi possible de calculer explicitement le Hessien de la fonction duale.

Il suffit pour cela, de dériver le gradient.

Nous avons ainsi :

$$\frac{\partial^2 \mathcal{L}}{\partial \lambda_i \partial \lambda_\ell} = \sum_{p=1}^{n} \frac{\partial \overline{f_i}}{\partial y_p} \frac{\partial y_p}{\partial \lambda_\ell} - \frac{\partial z_i}{\partial \lambda_\ell}$$

Comme nous connaissons, par ailleurs, la valeur des variables y_p en fonction des λ_i et de même pour les variables z_i, nous pouvons calculer les dérivées

$$\frac{\partial y_p}{\partial \lambda_\ell} \quad \text{et} \quad \frac{\partial z_i}{\partial \lambda_\ell}$$

et ainsi obtenir une expression du Hessien (Cfr. Annexe 6.D.).

Annexes au chapitre 6.

Annexe 6.A. : Forme séparable de la fonction duale.

$$\varphi(\lambda) = \min_{\substack{y_j \in D_j \; j=1\dots n \\ z_i \geq 0 \quad i=1\dots m}} \left[\overline{f}(y; x^k) + \sum_{i=1}^{m} (d_i z_i + d_i z_i^2) \right.$$

$$\left. + \sum_{i=1}^{m} \lambda_i \left(\overline{f_i}(y; x^k) - z_i - \hat{f}_i \right) \right]$$

$$= \min_{z_i \geq 0 \; i=1\dots m} \left(\sum_{i=1}^{m} (d_i z_i + d_i z_i^2) - \sum_{i=1}^{m} z_i \lambda_i \right)$$

$$+ \min_{y_j \in D_j \; j=1\dots n} \left(\overline{f}(y; x^k) + \sum_{i=1}^{m} \lambda_i \left(\overline{f_i}(y; x^k) - \hat{f}_i \right) \right)$$

Pour le premier terme :

$$\sum_{i=1}^{m} \min_{z_i \geq 0} \left(d_i z_i + d_i z_i^2 - z_i \lambda_i \right)$$

Pour le second terme :

$$\min_{y_j \in D_j \; j=1\dots n} \left[f(x^k) + \sum_{j \in J+} \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \frac{\partial f}{\partial y_j}\Big|_{x^k} \right.$$

$$+ \sum_{j \in J-} \left[(x_j^k - l_j^k) - \frac{(x_j^k - l_j^k)^2}{y_j - l_j^k} \right] \frac{\partial f}{\partial y_j}\Big|_{x^k}$$

$$+ \sum_{i=1}^{m} \lambda_i \left(f_i(x^k) + \sum_{j \in J_i+} \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right.$$

$$\left. \left. + \sum_{j \in J_i-} \left[(x_j^k - l_j^k) - \frac{(x_j^k - l_j^k)^2}{y_j - l_j^k} \right] \frac{\partial f_i}{\partial y_j}\Big|_{x^k} - \hat{f}_i \right) \right]$$

où $J+ = \{ j \in \{1,\dots,n\} \text{ tels que } \frac{\partial f}{\partial y_j}|_{x^k} \geq 0 \}$

$J- = \{ j \in \{1,\dots,n\} \text{ tels que } \frac{\partial f}{\partial y_j}|_{x^k} < 0 \}$

$Ji+ = \{ j \in \{1,\dots,n\} \text{ tels que } \frac{\partial f_i}{\partial y_j}|_{x^k} \geq 0 \}$

$Ji- = \{ j \in \{1,\dots,n\} \text{ tels que } \frac{\partial f_i}{\partial y_j}|_{x^k} < 0 \}$

$$= \min_{y_j \in D_j \; j=1\dots n} \Bigg[f(x^k) + \sum_{i=1}^{m} \lambda_i f_i(x^k) - \sum_{i=1}^{m} \lambda_i \hat{f_i}$$

$$+ \sum_{j \in J+} \Bigg(\left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \frac{\partial f}{\partial y_j}|_{x^k}$$

$$+ \sum_{i \in I_j+} \lambda_i \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}|_{x^k}$$

$$+ \sum_{i \in I_j-} \lambda_i \left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \frac{\partial f_i}{\partial y_j}|_{x^k} \Bigg)$$

$$+ \sum_{j \in J-} \Bigg(\left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \frac{\partial f}{\partial y_j}|_{x^k}$$

$$+ \sum_{i \in I_j+} \lambda_i \left[\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right] \frac{\partial f_i}{\partial y_j}|_{x^k}$$

$$+ \sum_{i \in I_j-} \lambda_i \left[(x_j^k - z_j^k) - \frac{(x_j^k - z_j^k)^2}{y_j - z_j^k} \right] \frac{\partial f_i}{\partial y_j}|_{x^k} \Bigg) \Bigg]$$

où $I_j+ = \{ i \in \{1,\dots,m\} \text{ tels que } \frac{\partial f_i}{\partial y_j}|_{x^k} \geq 0 \}$

$$I_j^- = \{ i \in \{1, \dots, m\} \text{ tels que } \frac{\partial f_i}{\partial y_j}\Big|_{x^k} < 0 \}$$

$$= f(x^k) + \sum_{i=1}^{m} \lambda_i f_i(x^k) - \sum_{i=1}^{m} \lambda_i \hat{f}_i + \sum_{j=1}^{n} \min_{y_j \in D_j} L_j(y_j; \lambda)$$

où $L_j(y_j; \lambda) =$

$$\text{si } j \in J^+ : \left(\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)$$

$$+ \sum_{i \in I_j^-} \lambda_i \left((x_j^k - \ell_j^k) - \frac{(x_j^k - \ell_j^k)^2}{y_j - \ell_j^k} \right) \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

$$\text{si } j \in J^- : \left((x_j^k - \ell_j^k) - \frac{(x_j^k - \ell_j^k)^2}{y_j - \ell_j^k} \right) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)$$

$$+ \sum_{i \in I_j^+} \lambda_i \left(\frac{(v_j^k - x_j^k)^2}{v_j^k - y_j} - (v_j^k - x_j^k) \right) \frac{\partial f_i}{\partial y_j}\Big|_{x^k}$$

Donc

$$\mathcal{L}(\lambda) = f(x^k) + \sum_{i=1}^{m} \lambda_i \left(f_i(x^k) - \hat{f}_i \right)$$

$$+ \sum_{i=1}^{m} \min_{z_i \geq 0} \left(d_i z_i + d_i z_i^2 - \lambda_i z_i \right)$$

$$+ \sum_{j=1}^{n} \min_{y_j \in D_j} L_j(y_j; \lambda)$$

Annexe 6.B.: Recherche des racines de $\frac{\partial L_j}{\partial y_j}$.

$$\frac{\partial L_j}{\partial y_j} = \frac{(v_j^k - x_j^k)^2}{(v_j^k - y_j)^2}\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ \sum_{i\in I_j-} \lambda_i \frac{(x_j^k - \ell_j^k)^2}{(y_j - \ell_j^k)^2} \frac{\partial f_i}{\partial y_j}\Big|_{x^k} = 0$$

$\Leftrightarrow$

$$(v_j^k - x_j^k)^2 (y_j - \ell_j^k)^2 \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ \sum_{i\in I_j-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} (x_j^k - \ell_j^k)^2 (v_j^k - y_j)^2 = 0$$

$\Leftrightarrow$

$$y_j^2 \left[(v_j^k - x_j^k)^2 \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right] + (x_j^k - \ell_j^k)^2 \left(\sum_{i\in I_j-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)\right]$$

$$+ y_j \left[-2\ell_j^k (v_j^k - x_j^k)^2 \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right] - 2 v_j^k (x_j^k - \ell_j^k)^2 \left(\sum_{i\in I_j-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)\right]$$

$$+ (\ell_j^k)^2 (v_j^k - x_j^k)^2 \left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ (v_j^k)^2 (x_j^k - \ell_j^k)^2 \left(\sum_{i\in I_j-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right) = 0$$

$$e = -4\,(v_j^k - x_j^k)^2\,(x_j^k - \ell_j^k)^2\,(v_j^k - \ell_j^k)^2$$

$$\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]\left[\sum_{i\in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

Voici les racines :

$$\Bigg(2\ell_j^k (v_j^k - x_j^k)^2\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)$$

$$+ 2 v_j^k (x_j^k - \ell_j^k)^2 \left(\sum_{i\in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)$$

$$\pm 2\,(v_j^k - x_j^k)(x_j^k - \ell_j^k)(v_j^k - \ell_j^k)$$

$$\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]^{1/2}\left[-\sum_{i\in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]^{1/2}\Bigg)\Big/$$

$$\Bigg(2\,(v_j^k - x_j^k)^2\left[\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]$$

$$+ 2\,(x_j^k - \ell_j^k)^2\left[\sum_{i\in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right]\Bigg)$$

ou encore :

$$(v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} z_j^k$$

$$\left[(v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} \mp (x_j^k - z_j^k)\left(-\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}\right]$$

$$+ v_j^k (x_j^k - z_j^k)\left(-\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}$$

$$\left[-(x_j^k - z_j^k)\left(-\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} \pm (v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}\right]$$

$$\Big/\left[(v_j^k - x_j^k)^2\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)\right.$$

$$\left. + (x_j^k - z_j^k)^2\left(\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)\right]$$

Annexe 6.C. : Rejet de l'une des racines.

Une des deux racines exprimées ci-dessus, peut encore s'écrire :

$$\frac{z_j^k (v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} - v_j^k (x_j^k - z_j^k)\left(-\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}}{(v_j^k - x_j^k)\left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i\in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2} - (x_j^k - z_j^k)\left(-\sum_{i\in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k}\right)^{1/2}}$$

c'est une racine qui ne nous convient pas car

elle n'appartient pas à l'intervalle :

$$\left[\max \{ \underline{x}_j , a_j^k \} \ , \ \min \{ \beta_j^k , \bar{x}_j \} \right]$$

En effet,

si son dénominateur est positif :

nous devrions avoir que cette racine est $> \zeta_j^k$

Mais alors :

$$\zeta_j^k < \left[\zeta_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right.$$

$$\left. - v_j^k (x_j^k - \zeta_j^k) \left(- \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right] /$$

$$\left[(v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right.$$

$$\left. - (x_j^k - \zeta_j^k) \left(- \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2} \right]$$

ou encore

$$\zeta_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2}$$

$$- v_j^k (x_j^k - \zeta_j^k) \left(- \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2}$$

$$> \zeta_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}\Big|_{x^k} + \sum_{i \in I_j^+} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2}$$

$$- \zeta_j^k (x_j^k - \zeta_j^k) \left(- \sum_{i \in I_j^-} \lambda_i \frac{\partial f_i}{\partial y_j}\Big|_{x^k} \right)^{1/2}$$

Ce qui nous donne : $v_j^k < l_j^k$

Or, par hypothèse, $l_j^k < v_j^k$

Donc cette racine est strictement supérieure à l_j^k

Si son dénominateur est négatif :
nous devrions avoir que cette racine est $< v_j^k$.

Mais alors :

$$v_j^k > \left[l_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2} \right.$$

$$\left. - v_j^k (x_j^k - l_j^k) \left(- \sum_{i \in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2} \right] /$$

$$\left[(v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2} \right.$$

$$\left. - (x_j^k - l_j^k) \left(- \sum_{i \in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2} \right]$$

ou encore

$$l_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2}$$

$$- v_j^k (x_j^k - l_j^k) \left(- \sum_{i \in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2}$$

$$> v_j^k (v_j^k - x_j^k) \left(\frac{\partial f}{\partial y_j}|_{x^k} + \sum_{i \in I_{j+}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2}$$

$$- v_j^k (x_j^k - l_j^k) \left(- \sum_{i \in I_{j-}} \lambda_i \frac{\partial f_i}{\partial y_j}|_{x^k} \right)^{1/2}$$

ζ qui nous donne : $L_j^k > U_j^k$ qui est absurde. Dans ce cas, la racine est strictement supérieure à U_j^k.

Annexe 6.D. : Hessien.

$$\frac{\partial^2 \mathcal{L}}{\partial \lambda_i \partial \lambda_\ell} = \sum_{p=1}^{m} \frac{\partial \overline{f_i}}{\partial y_p} \frac{\partial y_p}{\partial \lambda_\ell} - \frac{\partial z_i}{\partial \lambda_\ell}$$

où $$\frac{\partial z_i}{\partial \lambda_\ell} = \begin{cases} 0 & \text{si } i \neq \ell \\ \frac{1}{2 d_i} & \text{si } i = \ell \end{cases}$$

$$\frac{\partial \overline{f_i}}{\partial y_p} = \begin{cases} \frac{(U_p^k - x_p^k)^2}{(U_p^k - y_p)^2} \left.\frac{\partial f_i}{\partial y_p}\right|_{x^k} & \text{si } \left.\frac{\partial f_i}{\partial y_p}\right|_{x^k} \geq 0 \\ \frac{(x_p^k - L_p^k)^2}{(y_p - L_p^k)^2} \left.\frac{\partial f_i}{\partial y_p}\right|_{x^k} & \text{si } \left.\frac{\partial f_i}{\partial y_p}\right|_{x^k} < 0 \end{cases}$$

$$\frac{\partial y_p}{\partial \lambda_\ell} = ?$$

si $y_p = \max \{ \alpha_p, x_p^k \}$ ou $y_p = \min \{ \beta_p^k, \overline{x}_p \}$

alors $$\frac{\partial y_p}{\partial \lambda_\ell} = 0$$

sinon

si $\frac{\partial f}{\partial y_p}\Big|_{x^k} \geq 0$

il y a deux cas : si $\frac{\partial f_e}{\partial y_p}\Big|_{x^k} \geq 0$

$$\frac{\partial y_p}{\partial \lambda_e} = \left[\gamma_e \, (U_p^k - x_p^k)(x_p^k - L_p^k)(L_p^k - U_p^k) \right.$$

$$\left. \frac{\partial f_e}{\partial y_p}\Big|_{x^k} \left(\frac{\partial f}{\partial y_p}\Big|_{x^k} + \sum_{i \in I_p^+} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{-\gamma/2} \left(- \sum_{i \in I_p^-} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \right] /$$

$$\left[(U_p^k - x_p^k) \left(\frac{\partial f}{\partial y_p}\Big|_{x^k} + \sum_{i \in I_p^+} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \right.$$

$$\left. + (x_p^k - L_p^k) \left(- \sum_{i \in I_p^-} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \right]^2$$

si $\frac{\partial f_e}{\partial y_p}\Big|_{x^k} < 0$

$$\frac{\partial y_p}{\partial \lambda_e} = \left[\gamma/2 \, (x_p^k - L_p^k)(U_p^k - x_p^k)(L_p^k - U_p^k) \right.$$

$$\left. \frac{\partial f_e}{\partial y_p}\Big|_{x^k} \left(\frac{\partial f}{\partial y_p}\Big|_{x^k} + \sum_{i \in I_p^+} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \left(- \sum_{i \in I_p^-} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{-\gamma/2} \right] /$$

$$\left[(U_p^k - x_p^k) \left(\frac{\partial f}{\partial y_p}\Big|_{x^k} + \sum_{i \in I_p^+} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \right.$$

$$\left. + (x_p^k - L_p^k) \left(- \sum_{i \in I_p^-} \lambda_i \frac{\partial f_i}{\partial y_p}\Big|_{x^k} \right)^{\gamma/2} \right]^2$$

si $\frac{\partial f}{\partial y_p |x^k} < 0$

il y a deux cas : si $\frac{\partial f_\ell}{\partial y_p |x^k} \geq 0$

$$\frac{\partial y_p}{\partial \lambda_\ell} = \left[1/2 \, (u_p^k - x_p^k)(x_p^k - l_p^k)(l_p^k - u_p^k) \right.$$

$$\left. \frac{\partial f_\ell}{\partial y_p |x^k} \left(-\frac{\partial f}{\partial y_p |x^k} - \sum_{i \in I_{p^-}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \left(\sum_{i \in I_{p^+}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{-1/2} \right] /$$

$$\left[(x_p^k - l_p^k) \left(-\frac{\partial f}{\partial y_p |x^k} - \sum_{i \in I_{p^-}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \right.$$

$$\left. + (u_p^k - x_p^k) \left(\sum_{i \in I_{p^+}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \right]^2$$

si $\frac{\partial f_\ell}{\partial y_p |x^k} < 0$

$$\frac{\partial y_p}{\partial \lambda_\ell} = \left[1/2 \, (u_p^k - x_p^k)(x_p^k - l_p^k)(l_p^k - u_p^k) \right.$$

$$\left. \frac{\partial f_\ell}{\partial y_p |x^k} \left(-\frac{\partial f}{\partial y_p |x^k} - \sum_{i \in I_{p^-}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{-1/2} \left(\sum_{i \in I_{p^+}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \right] /$$

$$\left[(x_p^k - l_p^k) \left(-\frac{\partial f}{\partial y_p |x^k} - \sum_{i \in I_{p^-}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \right.$$

$$\left. + (u_p^k - x_p^k) \left(\sum_{i \in I_{p^+}} \lambda_i \frac{\partial f_i}{\partial y_p |x^k} \right)^{1/2} \right]^2$$

Chapitre 7 :

Algorithme .

§ 7.1 : Routine d'optimisation :

7.1.1 : Spécifications.

Nous avons voulu mettre sous forme de programme, la méthode de Svanberg.

Pour cela, nous avons considéré l'algorithme général. Et, pour des raisons précisées au chapitre 5, nous avons implémenté l'algorithme modifié que voici rappelé :

Pas 0 : Initialisation : - choix du point de départ
- poser $k = 0$

Pas 1 : Soit x^k.
Construction du sous-problème $\tilde{P}(x^k)$ associé à x^k.

Pas 2 : Résolution de $\tilde{P}(x^k)$.
Soit x^{k+1}, sa solution optimale.

Pas 3 : $k \leftarrow k+1$
Retourner au pas 1.

Dans un premier temps, étudions cette routine même d'optimisation.

Elle est capable de résoudre un problème du

type :

$$\begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f_i} \quad i = 1 \dots m \\ & \underline{x}_j \leq x_j \leq \bar{x}_j \quad j = 1 \dots n \end{cases}$$

Il faut noter que certaines conditions sont requises en ce sens que les bornes sur les variables doivent être explicites (les variables non bornées ou bornées d'un seul côté ne sont pas admises).

Cependant, aucune contrainte n'est imposée sur le point de départ qui peut donc être quelconque. La routine vérifie automatiquement s'il se pose des problèmes de domaine d'admissibilité vide pour un sous-problème, et, dans ce cas, ajuste les paramètres en conséquence. Vu l'emploi de variables artificielles, ces problèmes n'apparaissent que lorsque l'intervalle bornant les variables, est vide i.e. lorsque les intervalles $[\underline{x}_j, \bar{x}_j]$ et $[\alpha_j^k, \beta_j^k]$ n'ont pas d'intersection. Pour y remédier, s'il s'agit de bornes fictives, elles sont modifiées adéquatement. Sinon, nous bougeons le point de départ et le centrons dans $[\underline{x}_j, \bar{x}_j]$.

La résolution des sous-problèmes se fait

par le biais d'une routine tirée de la librairie NAG.

Il s'agit d'un algorithme de Quasi-Newton pour trouver le minimum d'une fonction à plusieurs variables, non soumise à des contraintes ou soumise seulement à des contraintes de bornes.

La routine doit aussi pouvoir connaître pour tout point, son image par la fonction objectif ou une fonction des contraintes. Elle doit aussi pouvoir évaluer les dérivées partielles de ces fonctions en un point donné quelconque.

Ce point peut paraître, à première vue, assez contraignant. En effet, au départ, les fonctions et leurs dérivées étaient fournies à la routine sous forme de sous-routines. Cependant, cela nécessitait plusieurs opérations de compilation et "link" avant chaque exécution. Mais cet inconvénient a pu être évité, comme cela sera expliqué plus loin, par l'intermédiaire d'un interpréteur de formules.

Le mode de construction des asymptotes associées au point de linéarisation, a été choisi et est le suivant. (Remarquons qu'il a été analysé au chapitre 1)

Pour $k = 1$ et $k = 2$:

$$\begin{cases} l_j^k = x_j^k - (\bar{x}_j - \underline{x}_j) \\ u_j^k = x_j^k + (\bar{x}_j - \underline{x}_j) \end{cases} \qquad j = 1 \dots n$$

Pour $k \geq 3$:

- s'il y a une oscillation

i.e. $(x_j^k - x_j^{k-1}) \cdot (x_j^{k-1} - x_j^{k-2}) < 0$

$$\begin{cases} l_j^k = x_j^k - s \cdot (x_j^{k-1} - l_j^{k-1}) \\ u_j^k = x_j^k + s \cdot (u_j^{k-1} - x_j^{k-1}) \end{cases} \qquad j = 1 \dots n$$

- s'il y a convergence monotone

i.e. $(x_j^k - x_j^{k-1}) \cdot (x_j^{k-1} - x_j^{k-2}) \geq 0$

$$\begin{cases} l_j^k = x_j^k - \dfrac{(x_j^{k-1} - l_j^{k-1})}{s} \\ u_j^k = x_j^k + \dfrac{(u_j^{k-1} - x_j^{k-1})}{s} \end{cases} \qquad j = 1 \dots n$$

où $s = 0.7$

Pour les "move-limits", nous avons suivi les suggestions de Svanberg [4] et avons ainsi posé : (Cfr. chapitre 3).

$$\begin{cases} a_j^k = 0.9 \; l_j^k - 0.1 \; x_j^k \\ \beta_j^k = 0.9 \; u_j^k + 0.1 \; x_j^k \end{cases} \qquad j = 1 \dots n$$

Quant aux conditions d'optimalité, nous avons mis celles de Kuhn-Tucker.

La routine fournit donc comme solution, un point admissible et vérifiant les conditions de Kuhn - Tucker.

Si donc, le domaine d'admissibilité du problème initial donné est vide, la routine ne fournira pas de solution et indiquera que le nombre maximal d'itérations a été atteint avant qu'elle n'ait réussi à trouver un point qui satisfait aux conditions d'optimalité.

7.2.1 : Algorithme.

L'algorithme décrivant ce que réalise la routine peut être schématisé comme suit :

Initialisation :

- des coéfficients des variables artificielles dans la fonction objectif (cfr. chapitre 5)
- des variables duales
- iter = 0

Etape générale :

Pas 1 : augmenter iter de 1

Pas 2 : vérifier si le nombre maximal d'itérations est atteint :

si oui : poser ERR = 1 et stopper

si non : aller au pas 3.

Pas 3 : évaluer les asymptotes mobiles (cfr. chapitre 1)

Pas 4 : évaluer les "move-limits" (cfr. chapitre 3)

Pas 5 : vérifier si le domaine admissible n'est pas vide :

si oui : ajuster les paramètres et aller au pas 3.

si non : aller au pas 6

Pas 6 : évaluer la fonction objectif et les contraintes en le point de linéarisation.

Pas 7 : initialisations pour la routine de NAG : E04KBF.

Pas 8 : résoudre le sous-problème par la routine de NAG.

Pas 9 : évaluer les variables primales et artificielles en la solution obtenue.

Pas 10 : vérifier si les conditions d'optimalité sont satisfaites :

si oui : poser ERR = 0

et stopper

si non : aller au pas 1

Remarque : nous rappelons que la résolution du sous-problème se fait par dualité, ce qui explique la nécessité du pas 9.
Nous signalons aussi qu'à la première itération, les variables duales sont initialisées par défaut à 1 et que pour les autres itérations, elles gardent les valeurs optimales obtenues à l'itération qui précède celle dont il est question.

§ 7.2 : "Habillement" informatique de la routine.

Nous avons ici essayé d'alléger quelque peu l'emploi de la routine décrite ci-dessus ainsi que d'élargir son domaine d'application.

Nous allons aborder les différentes

modifications qui y ont été apportées et les routines qui lui ont été associées.

7.2.1 : Les bornes inexistantes.

Tout d'abord, nous avons traité le problème des bornes sur les variables. En effet, dans la plupart des exemples ou des problèmes qu'il est intéressant de résoudre, nous n'avons pas toujours la possibilité de connaître les bornes qui doivent être mises sur les variables. Il est ennuyeux, alors, de ne pouvoir utiliser notre routine dans ces cas.

Il s'avère donc nécessaire de trouver un moyen pour pouvoir englober ces problèmes dans l'éventail de ceux que nous pouvons traiter.

C'est pourquoi nous avons implémenté une routine supplémentaire dont le but est de mettre automatiquement des bornes "fictives" là où il n'y en a pas. Cette routine fixe donc, elle-même, des bornes de façon arbitraire.

Nous avons convenu de fixer un intervalle de longueur 200 autour du point de départ donné pour suppléer les bornes inexistantes.

Bien sûr, nous tenons à garder la cohérence du

problème et si ces bornes "fictives" ne sont pas valables, nous posons :

borne inférieure = borne supérieure − 100

borne supérieure = borne inférieure + 100

Par exemple :

si nous avions $x \geq 200$,

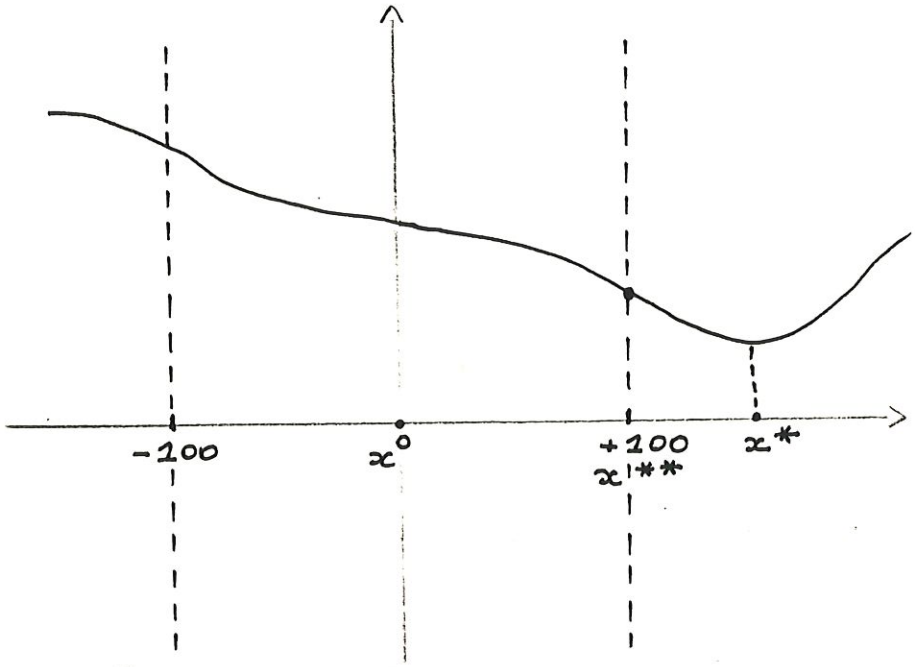
il est absurde de supposer $x \leq 100$.

Nous posons dans ce cas $x \leq 200 + 100 = 300$

Nous sommes alors capables de résoudre le problème ainsi complété.

N'oublions pas, cependant, qu'il nous faut pouvoir contrôler ces bornes "fictives" que nous imposons au processus.

En effet, examinons le schéma suivant :



Si nous supposons que la variable x n'est pas bornée, nous fixerons comme vu précédemment :

$$-100 \leq x \leq 100$$

Remarquons, sur le schéma, qu'en nous limitant à l'intervalle $[-100, +100]$, nous nous restreignons à une partie de la fonction.
Lors de la résolution, nous obtiendrons comme solution x^{**} qui est bien le minimum de la fonction sur l'intervalle imposé. Mais, rappelons-nous que cet intervalle a été fixé de manière arbitraire et il ne doit, en aucun cas, nous gêner pour la résolution. Or, il est clair qu'ici, l'une des bornes "fictives" étant active à l'optimum, ces contraintes fictives nous empêchent de poursuivre la minimisation.

Nous avons alors convenu que, lorsqu'à l'optimum, une variable atteint une borne dite "fictive", celle-ci est ajustée et nous recommençons à partir de cet optimum trouvé, la minimisation. Et ainsi de suite jusqu'à trouver un minimum qui n'atteint aucune borne fictive ou jusqu'à ce qu'un nombre total maximal d'itérations soit dépassé.

Dans l'exemple, nous allons reculer la borne

supérieure jusqu'à obtenir x^* comme minimum.

Ainsi ces bornes "fictives" que nous introduisons n'entravent pas le bon fonctionnement du processus de minimisation.

D'un point de vue pratique, nous avons construit une routine OPTSUP qui contrôle la résolution et s'occupe de la gestion des bornes "fictives".

Cette routine s'occupe aussi d'un autre problème qui s'est déclaré lors de tests. Nous avons, en effet, remarqué qu'un mauvais choix des bornes "fictives" pouvait entraîner un phénomène de cyclage qui perturbe la résolution. Si cela se produit, la routine OPTSUP ajuste les bornes "fictives" en réduisant l'intervalle i.e. en rapprochant les bornes. Nous reprenons, ensuite la minimisation avec les nouvelles bornes. Nous continuons de la sorte jusqu'à obtenir un minimum ou jusqu'à atteindre le nombre total maximal d'itérations imposé.

Nous avons ainsi réglé les problèmes d'utilisation et de gestion des bornes "fictives" que nous avons rencontrés au cours des tests effectués.

7.2.2 : l'interpréteur de formules.

D'un point de vue tout à fait différent, nous nous sommes préoccupés du problème de la disponibilité des données et principalement de la fonction objectif, des contraintes et de leurs dérivées respectives.

Dans un premier temps, nous disposions de ces données sous forme de routines qu'il était nécessaire de compiler et de "linker" avant chaque utilisation lorsque nous désirions changer de problème à résoudre.

C'était une perte de temps et c'était gênant pour l'utilisateur.

Nous avons voulu remédier à ce problème. C'est pourquoi, nous avons introduit un interpréteur de formules.

Ceci nous permet de rentrer les fonctions sous forme de fichier de textes.

L'utilisateur dispose d'un problème sous la forme "classique" :

$$\begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & f_i(x) \leq \hat{f}_i \quad i = 1 \dots m \\ & x \in X \quad \text{où} \quad X \subset \mathbb{R}^n \end{cases}$$

Il lui suffit de le rentrer tel quel dans un fichier. Il doit simplement le présenter sous la forme suivante :

```
MODEL
MIN = f(x)
fi(x) < f̂i
END
SLB   xi   xi
SUB   xp   x̄p
GUESS xl   3
```

Expliquons cette présentation et voyons quelle liberté est laissée quant à la forme exacte que doit avoir ce fichier.
Tout d'abord, il est possible d'introduire un problème de maximisation. Il faut pour cela, écrire MAX = f(x) au lieu de MIN = f(x).
Pour les contraintes, elles doivent nécessairement être sous la forme d'inégalités qui peuvent, cependant, être de 2 types (< ou >). Nous pouvons même envisager une combinaison de contraintes, les unes du type <, les autres du type >.
Par l'intermédiaire de l'expression SLB, nous

introduisons les bornes inférieures sur les variables. Il peut très bien ne pas y en avoir, ou bien y en avoir certaines et pas d'autres.
Le même principe est appliqué avec les bornes supérieures (SUB).
Toute variable ne doit donc pas être bornée ou peut l'être inférieurement sans l'être supérieurement et inversément.
'GUESS' nous permet de préciser le point de départ souhaité. Mais il est mis à 0 en cas d'absence. Et il nous est possible de spécifier une valeur pour certaines composantes, les autres étant initialisées à 0 par défaut.

Nous voyons ainsi que le procédé n'est pas trop rigide et qu'il permet de ne pas modifier l'expression du problème que nous avons à traiter vu qu'il envisage les diverses présentations possibles.

Analysons, maintenant, le fonctionnement de l'interpréteur.

Rôle : L'interpréteur lit une ligne du fichier et

l'analyse pour déterminer de quel type il s'agit. Il existe plusieurs types de lignes :

1) MODEL
2) GUESS
3) SLB
4) SUB

Suivant la manière dont il a classifié la ligne en question, il va la traiter de manière différente. Le cas le plus intéressant est, bien entendu, celui de la ligne "MODEL".

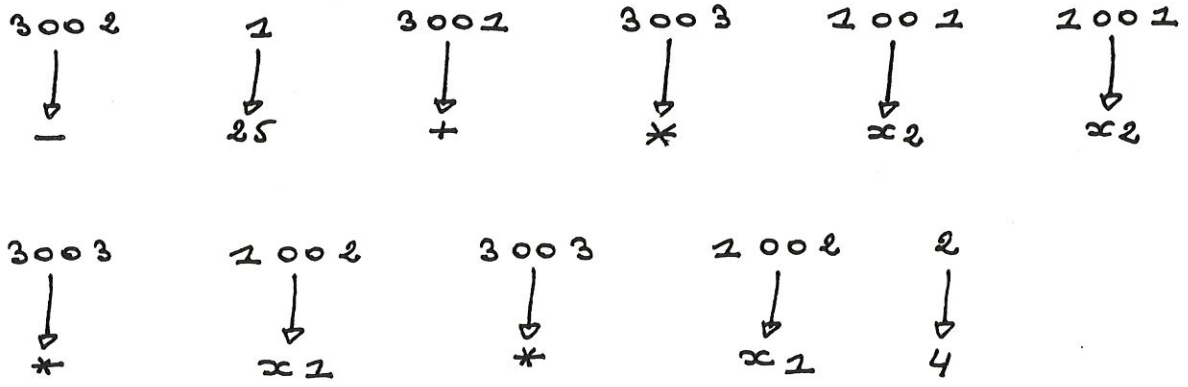
Dans ce cas, il lui faut "lire" la fonction objectif et les contraintes et les stocker en mémoire d'une manière telle qu'elles puissent être utilisées, ultérieurement par la routine d'optimisation proprement dite.

Ayant localisé l'expression de la fonction dans la ligne, il procède comme suit. Tout d'abord, il traduit cette expression en une formule codée sous forme de nombres désignant soit les opérations mathématiques effectuées, soit les variables ou les paramètres. Il stocke, ensuite, les variables et les paramètres dans une table de variables et une table de paramètres.

Voici un exemple :

$$4x_1^2 + x_2^2 - 25$$

Cette formule sera codée ainsi



La formule codée est alors stockée en mémoire.
Pour les contraintes, nous suivons le même schéma en ce qui concerne le membre de gauche de l'inégalité et nous stockons la valeur qui se trouve dans le membre de droite comme étant le second membre de la contrainte i.e. $\hat{fi}$.
L'interpréteur est capable d'adapter sa traduction suivant la spécificité de la ligne traitée. Ainsi s'il s'agit d'une maximisation, il traduira (-1) * (formule lue) au lieu de traduire simplement cette formule.
Un principe analogue est appliqué pour les contraintes du type >.
Remarquons l'équivalence des contraintes

$f_i(x) < a$ et $f_i(x) - a < 0$

(cfr. Annexe 7.A.)

Les deux présentations auront donc le même effet sur la résolution du problème.

Evaluations: Il paraît évidemment nécessaire de pouvoir se servir des formules codées fournies par l'interpréteur. C'est pourquoi, nous disposons d'une routine capable de les utiliser et d'évaluer la fonction représentée par cette formule en un point donné.

A chaque fois que nous avons besoin d'évaluer, soit la fonction objectif, soit une des contraintes, il nous suffit, alors, de faire appel à cette routine en précisant le point où se passe l'évaluation et la formule codée correspondant à la fonction évaluée.

En ce qui concerne l'évaluation des dérivées des fonctions, nous les calculons par différence finie en nous servant de la routine décrite précédemment qui évalue les fonctions.

7.2.3 : Le cas des contraintes d'égalité.

Nous avons voulu, ici, essayer d'élargir le domaine d'application de la méthode étudiée. Pour cela, nous avons envisagé des problèmes comprenant des contraintes d'égalité.

Ceci n'est, rappelons-nous, pas applicable si nous nous restreignons à la méthode décrite.

Mais il nous est possible de modifier quelque peu l'expression du problème en question :

$$\begin{cases} \text{minimiser} & f(x) \\ \text{s.c.} & h_i(x) = 0 \end{cases}$$

pour qu'il nous soit possible de le traiter tout de même.

Voici la transformation que nous avons considérée :

$$\begin{cases} \text{minimiser} & f(x) + \Pi s \\ \text{s.c.} & h_i(x) \leq s \\ & -h_i(x) \leq s \\ & s \geq 0 \end{cases}$$

h_i n'est pas convexe ? sauf si h_i linéaire

En donnant à Π une valeur suffisamment

grande pour forcer la variable s à atteindre sa borne inférieure 0, ces deux problèmes reviennent au même.

C'est pourquoi nous suppérons d'utiliser une telle transformation lorsque nous travaillons avec des problèmes comportant des contraintes sous forme d'égalité.

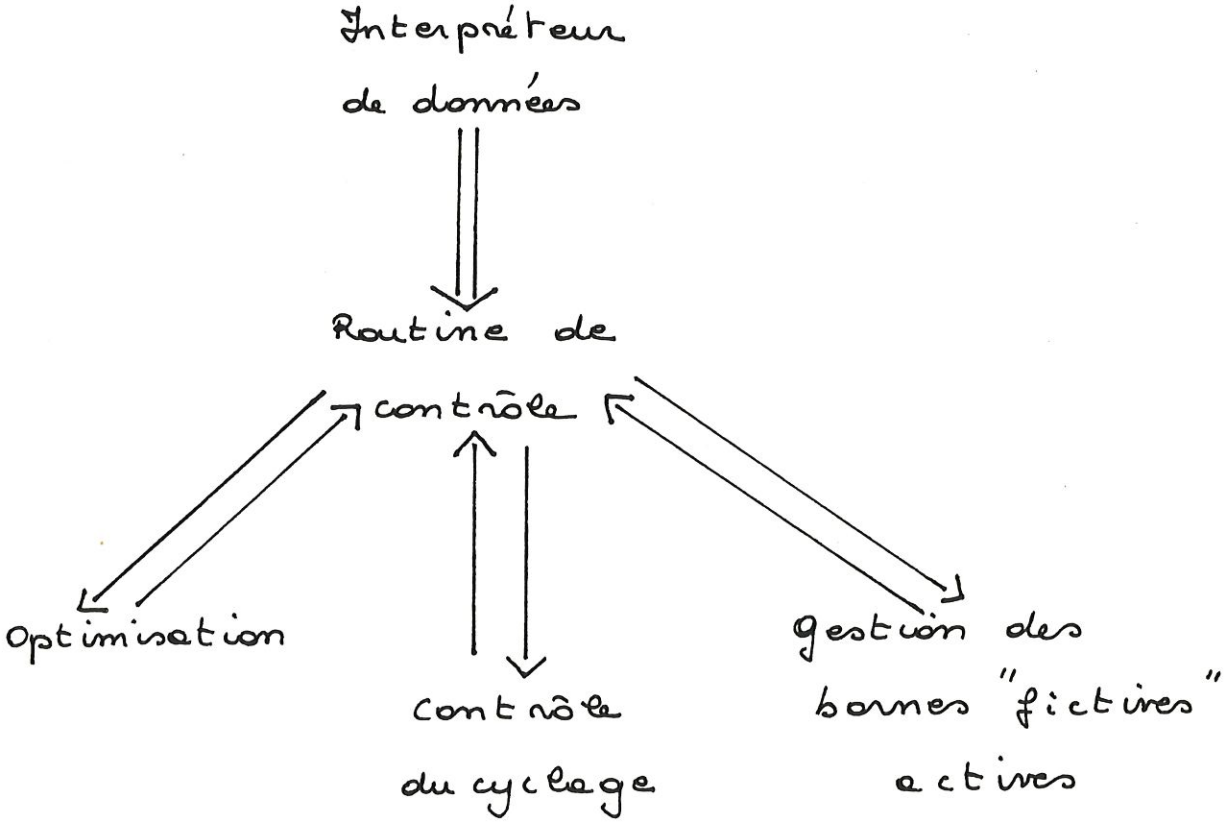
Nous n'avons pu effectuer beaucoup de tests à ce propos. Cependant, nous avons quand même pu remarquer le rôle joué par le paramètre n et l'influence qu'il a sur la vitesse de convergence du processus.

Nous avons repris quelques exemples étudiés plus loin en les modifiant pour leur ajouter des contraintes d'égalité. D'après les tests effectués et les résultats obtenus, cette façon de procéder semble être très satisfaisante. En effet, nous avons obtenu des convergences rapides.

Il faut considérer une valeur de n suffisamment grande sinon le processus risque de ne pas converger. Cependant, il ne faut pas non plus la prendre trop grande pour ne pas déséquilibrer le problème et perturber le processus.

7.2.4 : Conclusion.

Nous pouvons schématiser les modifications et les compléments apportés à notre routine comme suit :



L'utilisation des nouvelles routines nous évite bien des opérations de "link" et compilation et rend le programme plus aisé et agréable à manipuler.

Cela rend, également, le programme

indépendant et permet une utilisation plus externe, i.e. l'utilisateur ne doit plus se préoccuper de "programmation" proprement dite, ni ne doit plus "travailler" sur le problème par lui-même. Il lui suffit de répondre et d'introduire les données telles qu'elles se trouvent exprimées. Le programme se chargera de les façonner de manière à pouvoir les interpréter et se chargera aussi de veiller d'une part à suppléer les données manquantes, et d'autre part, à garder une certaine cohérence au problème considéré.

Les modifications apportées ont aussi comme conséquence un élargissement du domaine d'activité du programme. Nous avons, en effet, veillé à adapter ce programme de façon à pouvoir l'utiliser dans un grand nombre de cas très divers.

Annexe au chapitre 7.

Annexe 7.A.: Equivalence de contraintes.

Les contraintes :

$$f(y) - a \leq 0 \qquad \text{et} \qquad f(y) \leq a$$

sont équivalentes.

Il suffit de voir que ces deux contraintes vont se transformer après "linéarisation" de manière à produire des contraintes équivalentes dans chaque sous-problème.

La contrainte $f(y) - a \leq 0$ sera "linéarisée" autour de x comme suit :

$$f(y) - a \equiv z(y)$$

$$\overline{z}(y;x) = z(x) +$$

$$\sum_{j \in J+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial z}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J-} \left[(x_j - \ell_j) - \frac{(x_j - \ell_j)^2}{y_j - \ell_j} \right] \frac{\partial z}{\partial y_j}\Big|_x$$

où $J+ = \{ j \in \{1, \ldots, n\} \text{ tels que } \frac{\partial z}{\partial y_j}\Big|_x \geq 0 \}$

$$J- = \{ j \in \{1, \dots, n\} \text{ tels que } \frac{\partial z}{\partial y_j}\Big|_x < 0 \}$$

a $\frac{\partial z}{\partial y_j} = \frac{\partial f}{\partial y_j} \qquad j = 1 \dots n$

Donc $J+ = J'+ \qquad J- = J'-$

où

$$J'+ = \{ j \in \{1, \dots, n\} \text{ tels que } \frac{\partial f}{\partial y_j}\Big|_x \geq 0 \}$$

$$J'- = \{ j \in \{1, \dots, n\} \text{ tels que } \frac{\partial f}{\partial y_j}\Big|_x < 0 \}$$

Donc $\overline{z}(y ; x) = f(x) - a$

$$+ \sum_{j \in J'+} \left[\frac{(v_j - x_j)^2}{v_j - y_j} - (v_j - x_j) \right] \frac{\partial f}{\partial y_j}\Big|_x$$

$$+ \sum_{j \in J'-} \left[(x_j - \ell_j) - \frac{(x_j - \ell_j)^2}{y_j - \ell_j} \right] \frac{\partial f}{\partial y_j}\Big|_x$$

Et alors $\overline{z}(y ; x) = \overline{f}(y ; x) - a$

La contrainte $z(y) \leq 0$ deviendra alors :

$$\overline{z}(y ; x) = \overline{f}(y ; x) - a \leq 0$$

i.e. $\overline{f}(y ; x) \leq a$

Et nous obtenons alors la transformation de la seconde contrainte :

$$f(x) \leq a .$$

Les deux contraintes vont donc bien se transformer de façon à donner des contraintes équivalentes.

Chapitre 8 :

Point de vue numérique.

§ 8.1 : Tests.

Les différents problèmes que nous avons testés sont tirés des deux ouvrages suivants :

[9] : "Test Examples for Non Linear Programming Codes"
W. Hock K. Schittkowski
Lecture Notes in Economics and Mathematical Systems n° 187
1981 Springer-Verlag
Berlin Heidelberg New-York

[10] : "Elements of structural optimisation"
R.T. Haftka M.P. Kamat
Martinus Nij Hoff Publishers
1985

Voici les résultats que nous avons obtenus pour chacun d'entre eux.

Le nombre d'itérations donné est une information très relative puisqu'il dépend de la précision demandée (ici 10^{-6}).

Toutefois les indications données sur les différentes itérations permettent de se faire une idée sur le comportement de la méthode.

Problème n° 12 [9] :

$$\begin{cases} \text{minimiser} & 0.5\ x_1^2 + x_2^2 - x_1 x_2 - 7x_1 - 7x_2 \\ \text{s.c.} & 25 - 4x_1^2 - x_2^2 \geq 0 \end{cases}$$

Point de départ : $x^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Solution exacte : $x^* = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$

Valeur de la fonction objectif : $f(x^*) = -30.$

Le problème ne comporte aucune contrainte de borne sur les variables.
Le processus de mise automatique de bornes fictives a joué.
Aucun ajustement ne s'est avéré nécessaire.

Nombre d'itérations effectuées : 26

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	3600.000000000000
7	-8.9353508399999948
13	-26.784282590454 39
26	-30.000000000021 73

Remarquons que, jusqu'à l'itération 13, il se passe comme une période de "flottement" où le processus semble osciller.

Cela se voit encore plus clairement au niveau des itérés. En effet, le signe des composantes est alterné.

A partir de la 13e itération, le processus, par contre, se concentre autour de la solution. Nous entendons par là, qu'il a encore tendance à osciller autour de la solution mais que, maintenant, il se confine dans un petit voisinage autour de cette solution.

En résumé, nous pourrions dire que, dans un premier temps, il y a un essai de localisation du minimum de façon globale et ensuite,

dans la région déterminée, une recherche plus "fine" de la solution.

Problème n° 23 [9] :

$$\begin{cases} \text{minimiser} & x_1^2 + x_2^2 \\ \text{s.c.} & x_1 + x_2 - 1 \geq 0 \\ & x_1^2 + x_2^2 - 1 \geq 0 \\ & 9x_1^2 + x_2^2 - 9 \geq 0 \\ & x_1^2 - x_2 \geq 0 \\ & x_2^2 - x_1 \geq 0 \\ & -50 \leq x_i \leq 50 \quad i = 1, 2 \end{cases}$$

Point de départ : $x^0 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$

Solution exacte : $x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Valeur de la fonction objectif : $f(x^*) = 2$.

Les variables sont ici toutes bornées, il n'y a donc pas eu besoin d'ajouter des bornes fictives ni de les ajuster en cas de besoin.

Nombre d'itérations effectuées : 6

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	4.9618246618930 52
2	2.706912264348984
3	2.086113948645445
4	2.001811050466979
5	2.000000824585262
6	2.000000112743157

Contrairement au cas précédent, nous ne pouvons pas discerner ici, deux phases dans le processus.

Le point de départ et les bornes sont tels que nous nous trouvons de suite bien situés par rapport au minimum.

Et, comme cela sera analysé plus tard, le processus converge assez rapidement vers ce minimum.

Problème n° 66 [9] :

$$\begin{cases} \text{minimiser} & 0.2\, x_3 - 0.8\, x_1 \\ \text{s.c.} & x_2 - \exp(x_1) \geq 0 \\ & x_3 - \exp(x_2) \geq 0 \\ & 0 \leq x_1 \leq 100 \\ & 0 \leq x_2 \leq 100 \\ & 0 \leq x_3 \leq 10 \end{cases}$$

Point de départ : $x^0 = (0 \quad 1.05 \quad 2.9)^T$

Solution exacte :

$x^* = (0.18412648\,79 \quad 1.202167873 \quad 3.827322322)^T$

Valeur de la fonction objectif : $f(x^*) = 0.5181632741$

Nombre d'itérations effectuées : 11

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	0.4834959406683538
4	0.5181680506254076
8	0.5181632660113890
11	0.5181632740959337

De nouveau, le processus a pu déterminer de suite la région du minimum et y rester pour le trouver avec précision.

Une seule étape donc, apparaît ici pour la méthode.

Problème n° 72 [9] :

$$\begin{cases} \text{minimiser} \quad 1 + x_1 + x_2 + x_3 + x_4 \\ \text{s.c.} \quad 0.0401 - \dfrac{4}{x_1} - \dfrac{2.25}{x_2} - \dfrac{1}{x_3} - \dfrac{0.25}{x_4} \geq 0 \\ \qquad 0.0100085 - \dfrac{0.16}{x_1} - \dfrac{0.36}{x_2} - \dfrac{0.64}{x_3} - \dfrac{0.64}{x_4} \geq 0 \\ \qquad 0.001 \leq x_i \leq (5 - i) \in 5 \quad i = 1, \ldots, 4 \end{cases}$$

Point de départ : $x^0 = (1 \quad 1 \quad 1 \quad 1)^T$

Solution exacte :

$x^* = (193.4071 \quad 179.5475 \quad 185.0186 \quad 168.7062)^T$

Valeur de la fonction objectif : $f(x^*) = 727.67\,937$

Nombre d'itérations effectuées : 91

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	8.479730397260447
3	1.007312183947374
10	1.007418939113190
20	1.007418811559456
30	1.007419435143239
40	1.007441677218049
50	1.010026763471530
55	1.133992118986394
61	2.256010754388427
67	11.79301179478855
74	132.2298100640787
80	741.8016730316127
84	727.6341514599025
91	727.6793560174190

Ce problème présente un intérêt différent des précédents en ce sens que, pendant les 49 premières itérations, la méthode se concentre autour de 1.007... comme si le minimum s'y trouvait fort proche. Remarquons que le point vers lequel nous semblons nous diriger ne

satisfait pas les contraintes.

Ensuite, la méthode s'écarte d'abord lentement pendant les 15 itérations qui suivent et par après, plus rapidement pour aller se poster dans un voisinage du minimum réel.

Là, il se passe assez peu de temps pour déterminer avec précision ce minimum et après 5 itérations, la méthode en est déjà très proche.

Le processus semble donc d'abord minimiser la fonction et ensuite, rendre réalisable ce minimum trouvé, du moins dans cet exemple-ci.

Problème 76 [9] :

$$\begin{cases} \text{minimiser} & x_1^2 + 0.5\,x_2^2 + x_3^2 + 0.5\,x_4^2 \\ & - x_1 x_3 + x_3 x_4 - x_1 - 3x_2 + x_3 - x_4 \\ \text{s.c.} & 5 - x_1 - 2x_2 - x_3 - x_4 \geq 0 \\ & 4 - x_1 - x_2 - 2x_3 + x_4 \geq 0 \\ & x_2 + 4x_3 - 1.5 \geq 0 \\ & 0 \leq x_i \qquad i = 1, \ldots, 4 \end{cases}$$

Point de départ : $x^0 = (0.5 \quad 0.5 \quad 0.5 \quad 0.5)^T$

Il se produit au départ de petites oscillations qui se stabilisent rapidement autour du minimum recherché.

A partir de la 15e itération, nous sommes en possession d'une bonne approximation du résultat.

Problème n° 106 [9] :

$$
\begin{cases}
\text{minimiser} & x_1 + x_2 + x_3 \\
\text{s.c.} & 1 - 0.0025\,(x_4 + x_6) \geq 0 \\
& 1 - 0.0025\,(x_5 + x_7 - x_4) \geq 0 \\
& 1 - 0.01\,(x_8 - x_5) \geq 0 \\
& x_1 x_6 - 833.3325\, x_4 - 100 x_1 + 83333.333 \geq 0 \\
& x_2 x_7 - 1250\, x_5 - x_2 x_4 + 1250\, x_4 \geq 0 \\
& x_3 x_8 - 1250000 - x_3 x_5 + 2500\, x_5 \geq 0 \\
& 100 \leq x_1 \leq 10\,000 \\
& 1000 \leq x_i \leq 10000 \qquad i = 2, 3 \\
& 10 \leq x_i \leq 1000 \qquad i = 4, \ldots, 8
\end{cases}
$$

Point de départ :

$x^0 = (5000 \quad 5000 \quad 5000 \quad 200 \quad 350 \quad 150 \quad 225 \quad 425)^T$

Solution exacte : $x^* = (579.3167 \quad 1359.943 \quad 5110.072$

$182.0174 \quad 295.5985 \quad 227.9799 \quad 286.4162 \quad 395.5979)^T$

Valeur de la fonction objectif : $f(x^*) = 7049.330923$

Nombre d'itérations effectuées : 48

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	2100.000000000000
10	7047.600995453416
13	7049.747615649136
18	7049.248028680400
48	7049.248022031177

Nous obtenons jusqu'à la 10e itération, une période de localisation du minimum et à partir de là, en restant dans la région déterminée, nous recherchons avec grande précision le minimum.

Remarquons que la 18e itération est déjà une très bonne approximation.

Les suivantes ne servent qu'à gagner quelques décimales supplémentaires.

Problème n° 118 [9] :

$$
\begin{cases}
\text{minimiser} \quad \sum_{k=0}^{4} \left(2.3\, x_{3k+1} + 0.0001\, x_{3k+1}^2 + 1.7\, x_{3k+2} + 0.0001\, x_{3k+2}^2 + 2.2\, x_{3k+3} + 0.00015\, x_{3k+3}^2 \right) \\
\text{s.c.} \quad
\begin{cases}
\text{pour } j = 1 \ldots 4 : \\
\quad 0 \le x_{3j+1} - x_{3j-2} + 7 \le 13 \\
\quad 0 \le x_{3j+3} - x_{3j} + 7 \le 13 \\
x_1 + x_2 + x_3 - 60 \ge 0 \\
x_7 + x_8 + x_9 - 70 \ge 0 \\
x_{13} + x_{14} + x_{15} - 100 \ge 0 \\
\quad 0 \le x_{3j+2} - x_{3j-1} + 7 \le 14 \\
x_4 + x_5 + x_6 \ge 0 \\
x_{10} + x_{11} + x_{12} - 85 \ge 0 \\
8 \le x_1 \le 21 \\
43 \le x_2 \le 57 \\
3 \le x_3 \le 16 \\
\left.
\begin{array}{l}
0 \le x_{3k+1} \le 90 \\
0 \le x_{3k+2} \le 120 \\
0 \le x_{3k+3} \le 60
\end{array}
\right\} k = 1 \ldots 4
\end{cases}
\end{cases}
$$

Point de départ : $x^0 = (20 \quad 55 \quad 15 \quad 20 \quad 60 \quad 80 \quad 20 \quad 60 \quad 20 \quad 20 \quad 60 \quad 20 \quad 20 \quad 60 \quad 20)^T$

Solution exacte : $x^* = (8 \quad 49 \quad 3 \quad 1 \quad 56 \quad 0 \quad 1 \quad 63 \quad 6 \quad 3 \quad 70 \quad 12 \quad 5 \quad 77 \quad 18)^T$

Valeur de la fonction objectif : $f(x^*) = 664.8204500$

Nombre d'itérations effectuées : 12

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	748.9269988323367
5	665.3602879837163
9	664.9802403482738
12	664.8204503403138

Le nombre élevé de contraintes ne paraît pas être un obstacle à la résolution du problème.

Problème p. 121 [10] :

$$\begin{cases} \text{minimiser} & 3x_1 + \sqrt{3}\, x_2 \\ \text{s.c.} & 3 - \dfrac{18}{x_1} - \dfrac{6\sqrt{3}}{x_2} \geq 0 \\ & x_1 - 5.73 \geq 0 \\ & x_2 - 7.17 \geq 0 \end{cases}$$

Point de départ : $x^0 = (11.61 \quad 7.17)^T$

Solution exacte : $x^* = (9.464 \quad 9.464)^T$

Valeur de la fonction objectif : $f(x^*) = 44.784229$

Remarquons l'absence de bornes supérieures.

Nombre d'itérations effectuées : 13

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	37.11746410747393
7	44.77253191702913
8	44.78469660136762
13	44.78460872620047

Le processus converge rapidement vers la solution. La 8e itération donne déjà une très bonne approximation de ce minimum.

Problème p. 123 [10] :

$$\begin{cases} \text{minimiser} & x_1 + x_2 + x_3 \\ \text{s.c.} & 8 - x_1^2 - x_2^2 \geq 0 \\ & x_3 - 4 \geq 0 \\ & x_2 + 8 \geq 0 \end{cases}$$

Point de départ : $x^0 = (1 \quad 1 \quad 1)^T$

Solution exacte : $x^* = (-2 \quad -2 \quad 4)^T$

Valeur de la fonction objectif : $f(x^*) = 0$

Nombre d'itérations effectuées : 100 + 35

L'intérêt de ce problème est le suivant. Nous remarquons l'absence de bornes autour de la variable x_1 ainsi que de bornes supérieures sur x_2 et x_3. Elles sont donc fixées automatiquement et de façon arbitraire dans le processus.

Mais après 100 itérations, le nombre maximal d'itérations ayant été atteint sans

qu'un minimum n'ait été détecté, il s'est produit un ajustement des bornes fictives.

Cela a permis une convergence en 35 itérations qui ne se serait pas produite sinon, car nous avions détecté dans la première phase, un phénomène de cyclage de la méthode.

Pour la seconde phase, nous avons :

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	-31.030354198130 94
24	-5.6037241688434 72
35	-6.1161742337390 02E-11

Nous remarquons jusqu'à l'itération 24, certaines oscillations assez larges tandis qu'après, elles ne se produisent plus.

Problème p. 203 [10] :

$$\begin{cases} \text{minimiser} & x_1^2 + x_2^2 + x_3^2 \\ \text{s.c.} & 2 - \dfrac{16}{x_1^2} - \dfrac{16}{x_2^2} \geq 0 \\ & 2 - \dfrac{16}{x_2^2} - \dfrac{16}{x_3^2} \geq 0 \end{cases}$$

Point de départ : $x^0 = (2 \quad 2 \quad 2)^T$

Solution exacte : $x^* = (5.2262 \quad 6.2152 \quad 5.2262)^T$

Valeur de la fonction objectif : $f(x^*) = 93.253801$

Nombre d'itérations effectuées : 22

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	802.47263437373 24
12	80.874761385766 21
13	91.855290820729 40
21	93.254833995139 57

Jusqu'à la 12e itération, nous détectons

une phase de détermination approximative du minimum.

Il est à noter qu'ici, toutes les bornes sur les variables sont fictives et donc fixées arbitrairement.

Aucun ajustement ne s'est avéré nécessaire.

Problème p. 123 [10] :

$$\begin{cases} \text{minimiser} & x_1 + 2x_2 \\ \text{s.c.} & 1 - \dfrac{1.5}{x_2 + 0.25 x_1} \geq 0 \\ & x_1 \geq 0.5 \\ & x_2 \geq 0.5 \end{cases}$$

Point de départ : $x^0 = (1 \quad 1)^T$

Solution exacte : $x^* = (0.5 \quad 1.375)^T$

Valeur de la fonction objectif : $f(x^*) = 3$.

Nombre d'itérations effectuées : 4

Remarquons la rapidité de la convergence du processus.

ITERATION	VALEUR DE LA FONCTION OBJECTIF
1	3.170178699920030
2	3.247906378204527
3	3.249998554558983
4	3.250000000082213

§ 8.2 : Conclusions.

Une première constatation d'ordre général peut être tirée des tests effectués.

La méthode semble être assez sensible aux bornes imposées sur les variables et ce phénomène peut influencer assez fortement la vitesse avec laquelle le processus converge vers la solution et donc ainsi, le nombre d'itérations.

Ceci pourrait en partie s'expliquer par les asymptotes mobiles qui sont construites au départ en fonction des bornes. Si donc,

elles sont mal initialisées vis-à-vis du problème, le processus mettra plus de temps pour arriver au minimum. Le problème consisterait, alors, à déterminer pour chaque cas, en particulier, les bornes les plus adéquates.

Voici deux exemples de ce phénomène :

Problème n° 12 (cfr. p. 8.2) :

Comme vu et analysé précédemment, sans bornes et donc en utilisant le dispositif de mise automatique de bornes sur les variables, le processus met 16 itérations pour atteindre le minimum.

D'autre part, si nous exigeons la même précision mais imposons des bornes au problème, nous obtenons un résultat différent.

Voici les bornes que nous avons choisies :

$$-1 \leq x_1 \leq 4$$

$$-1 \leq x_2 \leq 4$$

Nous arrivons alors au minimum en 9 itérations.

Remarquons que la différence entre les deux est assez significative.

Voici un autre exemple très particulier.

Problème 122 (cfr. p. 8.15) :

Nous avons fixé les bornes supérieures à différentes valeurs et nous avons obtenu :

- Si elles valent 10 14 itérations
- Si elles valent 20 6 itérations
- Si nous les laissons indéfinies 13 itérations

Ce cas peut paraître surprenant. En effet, nous pourions croire que pour diminuer le nombre d'itérations, il suffit de rapprocher les bornes. Or, ce test nous fournit la preuve que ce n'est pas toujours le cas.

Ici, en augmentant l'intervalle de variation, nous avons diminué de moitié, le nombre d'itérations.

Il paraît donc assez difficile d'établir des bornes permettant de minimiser le nombre d'itérations.

Il faut cependant remarquer que, même dans les cas où le pire choix fut effectué, la convergence est assez rapide et le processus se comporte bien.

Notre deuxième constatation concerne un essai d'analyse pour déterminer une vitesse de convergence.

Dans ce cadre, nous nous sommes attachés à certains problèmes testés en particulier.

Problème 213 (cfr p. 8.19):

Nous remarquons, dans ce cas, une convergence "quadratique".

Il nous faut préciser que pour établir ceci, nous avons calculé la différence en valeur absolue de la valeur de la fonction objectif en la solution et en chaque itéré, ainsi que la différence relative (i.e. valeur précédente divisée en valeur absolue par la valeur de la fonction objectif en l'optimum).

Problème 203 (cfr p. 8.18):

Pendant la phase que nous avons appelée "localisation de la région du minimum", la différence relative diminue régulièrement.

Mais dans la seconde phase, lorsque le processus a atteint la zone en question, la

convergence devient linéaire.

Problème n°66 (cfr p. 8.6):

Nous remarquons, ici, de suite une convergence linéaire de la méthode.

Rappelons-nous avoir dit précédemment, que ce problème ne comportait qu'une phase. Le processus est, en effet, parti d'un endroit proche du minimum.

Problème n°83 (cfr. p. 8.4):

La vitesse de convergence est "quadratique".

D'autres problèmes ont aussi été testés à ce propos. Il en ressort, que lorsqu'il a déterminé une région voisine du minimum, la convergence devient parfois linéaire ou même quadratique. Mais ceci, bien sûr, n'est qu'une conclusion qui se rapporte aux analyses effectuées.

Conclusion :

Essentiellement, nous avons travaillé sur deux plans en parallèle.

Premièrement, nous avons repris la méthode de K. Svanberg. Nous l'avons analysée puis complétée par une recherche personnelle concernant la convergence. Nous avons introduit, pour obtenir des résultats de convergence relativement fins, une recherche linéaire exacte.

En parallèle, nous avons implémenté la méthode et avons construit, autour de cette routine, un environnement informatique selon les normes en usage chez SOLVAY, afin de rendre le logiciel plus convivial.

Cet environnement consiste, entre autres, en un interpréteur de formules et en des routines de contrôle.

Dans une perspective d'avenir, il serait intéressant, par exemple, d'approfondir le cas des contraintes d'égalité, et d'envisager l'introduction de variables discrètes, vu les applications en structure mécanique auxquelles est destinée la méthode étudiée.

Annexe :

Programme

guide de l'utilisateur :

Pour pouvoir se servir du programme, l'utilisateur doit simplement, comme spécifié dans le chapitre 7, disposer d'un fichier de texte contenant le problème à traiter.

Ce fichier doit être construit comme suit :

* Le fichier a comme première ligne :

  MODEL

* Ensuite, vient la fonction objectif. Pour cela, nous précisons s'il s'agit d'une minimisation (MIN) ou d'une maximisation (MAX).

  Par exemple :

  $$MIN = x_1 * x_2 + x_2$$

  ou $$MAX = 3 * x_1 - x_2$$

* Les lignes suivantes contiennent les contraintes qui sont nécessairement des inégalités qui peuvent, cependant, être du type $\leq$ ou du type $\geq$.

  Par exemple :

  $$x_2 + 3 * x_3 > 4$$

  $$x_3 - 4 * x_2 + 3 < 0$$

Les signes sont des inégalités strictes alors qu'ils correspondent, en réalité, à des inégalités non strictes. Mais ces signes ne servent, en fait, qu'à indiquer de quel type de contraintes il s'agit.

Remarquons que le second membre de l'inégalité doit être un paramètre. Les variables doivent donc toutes figurer dans le premier membre.

× Cette partie est close par le mot : END.

× Viennent, ensuite, les contraintes de bornes s'il en existe.

Nous précisons, tout d'abord, le type de bornes (borne inférieure : SLB ; borne supérieure : SUB), ensuite la variable dont il est question et enfin, la valeur de la borne.

Par exemple :

SLB x 1 3

ou SUB x 3 4

× Pour terminer, nous spécifions le point de départ.

Par exemple :

GUESS x 1 3

si la variable x_1 est initialisée à 3.
Les variables sont initialisées à 0 par défaut.

En résumé, nous aurons un fichier du type :

```
MODEL
MIN = f(x1, x2)
f1 (x1, x2) < 3
f2 (x1, x2) < 0
END
SLB    x1    3
SUB    x2    4
GUESS    x1    0
GUESS    x2    3
```

Il suffit alors de "faire tourner" le programme.

Celui-ci demande si les résultats doivent être donnés à l'écran ou sur fichier. Si cette dernière proposition est choisie, l'utilisateur sera invité à préciser le nom du fichier qu'il désire.
Le programme, enfin, s'enquiert du nom du fichier contenant l'énoncé du problème.
Les résultats intermédiaires et finaux sont alors

mis à la disposition de l'utilisateur.

Nous voyons donc que l'utilisation de notre programme a été rendue très simple et est ainsi à la portée de toute personne.

```
C     ************************************************************************
      PROGRAM MEMOIRE
C     ===============
C
C     Ce programme resoud un probleme de minimisation sous contraintes en
C     utilisant la methode de Svanberg.
C     Ceci a ete realise par Vinciane De Baets dans le cadre de son memoire
C     en seconde licence mathematique.
C
C     Documentation:
C     -------------
C       Variables du programme:
C       ----------------------
C       ITMAX : nombre maximal d'iterations permises lors d'une minimisation
C               particuliere (INT)
C       ERR : indique si le processus s'est deroule avec succes (ERR=0) ,
C             si le nombre maximal total d'iterations a ete atteint (ERR=2) ,
C             ou encore si le nombre maximal d'iterations maximal a ete
C             atteint lors d'une minimisation et qu'il n'est pas possible
C             de modifier un parametre en vue de traiter une autre
C             minimisation (INT)
C       IT : nombre d'iterations effectue (INT)
C       ITTOT : nombre d'iterations maximal pour le processus general (INT)
C       EPS : precision souhaitee (DP)
C       SORTIE : contient le nom du fichier devant contenir les resultats
C                (CHAR)
C       CHOIX : indique si l'utilisateur desire ses resultats a l'ecran (E)
C               ou sur un fichier (CHAR)
C
C       Parametres du programme:
C       -----------------------
C       MAXCON : nombre maximal de contraintes
C       MAXVAR : nombre maximal de variables
C
C       Commons:
C       -------
C       INT : ITER : numero de l'iteration (INT)
C             NV : nombre de variables primales (INT)
C             P : nombre de contraintes (INT)
C       CDON1 : BINF : tableau contenant la borne inferieure sur X (DP)
C               BSUP : tableau contenant la borne superieure sur X (DP)
C       DON2 : X : tableau contenant le point de linearisation Xk et
C                  contenant , a l'entree , les variables primales
C                  evaluees en fonction des anciennes variables duales
C                  et a la sortie , les variables primales evaluees en
C                  fonction des nouvelles variables duales (DP)
C              FCHAP : tableau contenant les seconds membres des
C                      contraintes (DP)
C              Z : tableau contenant , a l'entree , les variables
C                  artificielles evaluees en fonction des anciennes
C                  variables duales et a la sortie , les variables
C                  artificielles evaluees en fonction des nouvelles
C                  variables duales (DP)
C
```

```
C
C       Declarations:
C       ------------
C
        INTEGER NV,P,ITER,MAXCON,MAXVAR,ITMAX,ERR,IT,ITTOT
        PARAMETER(MAXCON=100)
        PARAMETER(MAXVAR=100)
        DOUBLE PRECISION X,Z,BINF,BSUP,FCHAP,EPS
        CHARACTER CHOIX
        CHARACTER*60 SORTIE,FICH
        DIMENSION BINF(MAXVAR),BSUP(MAXVAR)
        COMMON/INT/ITER,NV,P
      COMMON          /CDON1/BINF, BSUP
        COMMON/DON2/X(2*MAXVAR),FCHAP(MAXCON),Z(MAXCON)
C
C       Programme:
C       ---------
C
        EPS=1.E-3
        ITTOT=200
C
C       Lecture des donnees du probleme.
C
C
        CALL INTRO(CHOIX,SORTIE)
        CALL INIT(X,BINF,BSUP,MAXVAR)
        WRITE(5,*)'Quel est le nom du fichier contenant le probleme ?'
        READ(5,'(A60)') FICH
        OPEN(UNIT=1,FILE=FICH,STATUS='OLD')
        CALL PRFCN(NV,P)
        CLOSE(1)
        IF(CHOIX.EQ.'E')THEN
          SORTIE='TT:'
        ENDIF
101     OPEN(UNIT=27,FILE=SORTIE,STATUS='NEW',ERR=1100)
        GO TO 20
1100    WRITE(5,*)' Erreur a l''ouverture du fichier des resultats.'
        WRITE(5,*)' Veuillez donner un autre fichier s.v.p.'
        READ(5,*)SORTIE
        GO TO 101
20      CONTINUE
C
C       Optimisation.
C
        CALL OPTSUP(EPS,BINF,BSUP,ERR,IT,ITTOT)
C
C       Impression des resultats.
C
        IF(ERR.EQ.2.OR.ERR.EQ.1)THEN
          WRITE(27,*)
          WRITE(27,*)'NON CONVERGENCE DE LA METHODE'
          WRITE(27,*)'Le nombre maximal d''iterations est atteint.'
          WRITE(27,*)
        ENDIF
        IF(ERR.EQ.0)THEN
          CALL RESULT(ITER,X(1+MOD(ITER,2)*NV),Z,NV,P)
        ENDIF
        CLOSE(27)
        STOP
        END
C
C       ********************************************************************
```

```
C       ************************************************************************
C
        SUBROUTINE INIT(X,BINF,BSUP,DIM)
C       ================================
C
C       Role : La routine initialise le point de depart ainsi que les bornes
C       ----   sur la variable X.
C
C       Documentation:
C       -------------
C         Parametres d'entree:
C         -------------------
C         DIM : dimension maximale (INT)
C
C         Parametres de sortie:
C         --------------------
C         X : tableau de longueur egale au nombre de variables maximal ,
C             contenant le point de depart initialise par defaut (DP)
C         BINF : tableau de longueur egale au nombre de variables maximal ,
C                contenant les bornes inferieures initialisees par defaut (DP)
C         BSUP : tableau de longueur egale au nombre de variables maximal ,
C                contenant les bornes superieures initialisees par defaut (DP)
C
C
C       Specifications:
C       --------------
        INTEGER DIM
        DOUBLE PRECISION X,BINF,BSUP
        DIMENSION X(DIM),BINF(DIM),BSUP(DIM)
C
C       Declarations:
C       ------------
C
        INTEGER IND
C
C       Programme:
C       ---------
C
        DO 100 IND=1,DIM
         X(IND)=0.
100     CONTINUE
        DO 200 IND=1,DIM
         BINF(IND)=-10.E10
         BSUP(IND)=10.E10
200     CONTINUE
        RETURN
        END
C
```

```
C       ************************************************************************
C
        SUBROUTINE INTERM(ITER,XK,Z,N,P)
C       ================================================
C
C       Role : impression des resultats intermediaires.
C       ----
C
C       Documentation:
C       -------------
C         Parametres d'entree:
C         -------------------
C         ITER : indique l'iteration a laquelle on se trouve (INT)
C         XK : tableau de longueur egale au nombre de variables contenant ,
C              le minimum intermediaire (DP)
C         Z : tableau de longueur egale au nombre de contraintes , contenant
C             les variables artificielles intermediaires (DP)
C         N : nombre de variables (INT)
C         P : nombre de contraintes (INT)
C
C       Common:
C         ------
C         CVAR : NBVLIS : nombre de variables (INT)
C                VARLIS : table contenant les differentes variables (CHAR)
C
C
C       Specifications:
C       --------------
C
        INTEGER N,P,ITER
        DOUBLE PRECISION XK,Z,AUX
        DIMENSION XK(N)
C
C       Declarations:
C       ------------
C
        INTEGER K,NBVLIS,MAXLEN,LENVAR
        PARAMETER (MAXLEN=255,LENVAR=9)
        CHARACTER VARLIS(MAXLEN)* (LENVAR)
C
C       Common:
C       ------
C
        COMMON/CVAR/NBVLIS, VARLIS
C
C       Programme:
C       ---------
C
        WRITE(27,*)
        WRITE(27,*)' ITERATION:',ITER
        WRITE(27,*)' -----------'
        WRITE(27,*)
        CALL OBJ(XK,AUX,N)
        WRITE(27,*)' Valeur de la fonction objectif',AUX
        WRITE(27,*)' Variable :'
        DO 200 K=1,N
          WRITE(27,110)VARLIS(K),XK(K)
200     CONTINUE
110     FORMAT(11X,'composante ',A9,2x,F13.7)
        RETURN
        END
C
C       ************************************************************************
```

```
C     ********************************************************************************
C
      SUBROUTINE INTRO(CHOIX,SORTIE)
C     ==============================
C
C     Role : presentation du probleme.
C     ----
C
C     Documentation:
C     -------------
C       Parametres de sortie:
C       --------------------
C       CHOIX : indique si l'utilisateur desire les resultats sur fichier
C               (F) ou a l'ecran (E) (CHAR)
C       SORTIE : contient le nom du fichier des resultats (CHAR)
C
C
C     Specifications:
C     --------------
C
      CHARACTER CHOIX
      CHARACTER*60 FICH,SORTIE
C
C     Programme:
C     ---------
C
      WRITE(5,*)' MEMOIRE  1986-1987 .'
      WRITE(5,*)
      WRITE(5,*)' Ce programme resoud un programme non lineaire.'
      WRITE(5,*)' Il utilise la methode des "asymptotes mobiles."'
      WRITE(5,*)' Le  programme non lineaire est du type : '
      WRITE(5,*)
      WRITE(5,*)'  minimiser f(x)'
      WRITE(5,*)'  sous les contraintes :'
      WRITE(5,*)'     fi(x) <= f^i          1 <= i <= m'
      WRITE(5,*)'     xinf <= x <= xsup     ou x est dans Rn.'
      WRITE(5,*)
      WRITE(5,*)
      WRITE(5,*)' Desirez-vous les resultats a l''ecran ou sur fichier
      1 ? (F ou E).'
      READ(5,'(A)')CHOIX
      IF(CHOIX.NE.'E')THEN
        WRITE(5,*)' Quel est le nom du fichier ?'
        READ(5,'(A)')SORTIE
      ENDIF
      WRITE(5,*)
      RETURN
      END
C
```

```
C     ************************************************************************
C
      SUBROUTINE RESULT(ITER,XK,Z,N,P)
C     ================================
C
C     Role : impression des resultats.
C     ----
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       ITER : indique le nombre d'iterations effectuees lors de la derniere
C              minimisation (INT)
C       XK : tableau de longueur egale au nombre de variables , contenant
C            le minimum trouve (DP)
C       Z : tableau de longueur egale au nombre de contraintes , contenant
C           les variables artificielles optimales (DP)
C      N : nombre de variables (INT)
C       P : nombre de contraintes (INT)
C
C      Common:
C       ------
C       CVAR : NBVLIS : nombre de variables (INT)
C              VARLIS : table contenant les differentes variables (CHAR)
C
```

```
C
C       Specifications:
C       ---------------
C
        INTEGER N,P,ITER
        DOUBLE PRECISION XK,Z,AUX
        LOGICAL VERIF
        DIMENSION XK(N),Z(P)
C
C       Declarations:
C       -------------
C
        INTEGER K,NBVLIS,LENVAR,MAXLEN
        PARAMETER (MAXLEN=255,LENVAR=9)
        CHARACTER VARLIS(MAXLEN)* (LENVAR)
C
C       Common:
C       -------
C
        COMMON/CVAR/NBVLIS, VARLIS
C
C       Programme:
C       ----------
C
        WRITE(27,*)
        WRITE(27,*)' RESULTATS :'
        WRITE(27,*)' -----------'
        WRITE(27,*)
        WRITE(27,*)' Le nombre d''iterations effectuees est:',ITER
        WRITE(27,*)
        CALL OBJ(XK,AUX,N)
        WRITE(27,*)' Valeur de la fonction objectif',AUX
        WRITE(27,*)' Variable :'
        DO 200 K=1,N
          WRITE(27,110)VARLIS(K),XK(K)
200     CONTINUE
        VERIF=.TRUE.
        K=1
33      IF(Z(K).GT.0.)THEN
          VERIF=.FALSE.
          GO TO 22
        ELSE
          K=K+1
          IF(K.GT.P)GO TO 22
          GO TO 33
        ENDIF
22      IF(VERIF.EQ..FALSE.)THEN
          WRITE(27,*)'SOLUTION NON ADMISSIBLE'
        ENDIF
        WRITE(27,*)' Variable z :'
        DO 300 K=1,P
          WRITE(27,120)K,Z(K)
300     CONTINUE
110     FORMAT(11X,'composante ',A9,2x,F13.7)
120     FORMAT(11X,'composante no-',I3,2x,F13.7)
        RETURN
        END
C
```

1e Partie :

- Routine d'optimisation OPT
- Routines et fonctions utilisées par OPT.

```
C     ************************************************************************
C
      SUBROUTINE OPT(EPS,ERR,ITMAX)
      =============================

C
C     Role:routine d'optimisation.
C     ----
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       EPS : precision souhaitee (DP)
C       ITMAX : nombre maximal d'iterations autorisees (INT)
C
C       Parametres de sortie:
C       --------------------
C       ERR : indique si la minimisation s'est effectuee avec succes
C             (ERR=0) ou si le nombre maximal d'iterations a ete atteint
C             (ERR=1) (INT)
C
C       Commons:
C       -------
C       INT : ITER : numero de l'iteration (INT)
C             NV : nombre de variables primales (INT)
C             P : nombre de contraintes (INT)
C       AUX : L : tableau contenant les composantes de l'asymptote L
C                 a l'iteration presente (DP)
C             U : tableau contenant les composantes de l'asymptote U
C                 a l'iteration presente (DP)
C             A : tableau contenant les composantes de la "move-limit"
C                 alpha a l'iteration presente (DP)
C             B : tableau contenant les composantes de la "move-limit"
C                 beta a l'iteration presente (DP)
C             FCT : tableau contenant la fonction objectif et les
C                   contraintes evaluees en le point de linearisation Xk
C                   (DP)
C             DFJX : tableau contenant les derivees de la fonction
C                    objectif evaluees en Xk , point de linearisation (DP)
C             DFIJX : tableau dont la I-ieme ligne contient les derivees
C                     de la I-ieme contrainte evaluees en le point de
C                       linearisation Xk (DP)
C             D : tableau contenant les coefficients Di des variables
C                 artificielles dans la fonction objectif du probleme
C                 linearise (DP)
C       DON1 : XINF : tableau contenant la borne inferieure sur X (DP)
C              XSUP : tableau contenant la borne superieure sur X (DP)
C       DON2 : X : tableau contenant le point de linearisation Xk et
C                  contenant , a l'entree , les variables primales
C                  evaluees en fonction des anciennes variables duales
C                  et a la sortie , les variables primales evaluees en
C                  fonction des nouvelles variables duales (DP)
C                FCHAP : tableau contenant les seconds membres des
C                        contraintes (DP)
C                Z : tableau contenant , a l'entree , les variables
C                    artificielles evaluees en fonction des anciennes
C                    variables duales et a la sortie , les variables
C                    artificielles evaluees en fonction des nouvelles
C                    variables duales (DP)
```

```
C
C         Specifications:
C         --------------
C
          INTEGER NV,P,ERR,ITMAX,ITER
          DOUBLE PRECISION EPS,XINF,XSUP,FCHAP
C
C         Declarations:
C         ------------
C
          INTEGER MAXCON,MAXVAR
          PARAMETER(MAXCON=100)
          PARAMETER(MAXVAR=100)
          INTEGER LIW,LW,IW(2),IND,ISTATE,IFAIL,N,IPRINT,INTYPE,
         1        MAXCAL,IBOUND,LH
          DOUBLE PRECISION X,Z,L,U,SIGNE,S,A,B,FCT,DFJX,DFIJX,LAMBDA,
         1                 D,ETA,BL,BU,HESL,HESD,G,W,F,XTOL,FEST,
         2                 STEPMX
          LOGICAL TSTOPT,LOCSCH,MODIF
          COMMON/INT/ITER,NV,P
          COMMON/AUX/L(MAXVAR),U(MAXVAR),A(MAXVAR),B(MAXVAR),
         1             FCT(MAXCON+1),DFJX(MAXVAR),DFIJX(MAXCON,MAXVAR),
         2             D(MAXCON)
          COMMON/DON1/XINF(MAXVAR),XSUP(MAXVAR)
          COMMON/DON2/X(2*MAXVAR),FCHAP(MAXCON),Z(MAXCON)
          DIMENSION SIGNE(MAXVAR),LAMBDA(MAXCON),BL(MAXCON),BU(MAXCON),
         1          HESL(MAXCON*MAXCON/2),HESD(MAXCON),G(MAXCON),
         2          W(9*MAXCON),ISTATE(MAXCON)
          EXTERNAL MONIT,FUNCT,E04LBS
C
C         Programme:
C         ---------
C
C         Initialisation:
C
          DO 100 IND=1,P
            D(IND)=200000.
100       CONTINUE
          DO 200 IND=1,P
            LAMBDA(IND)=1.
200       CONTINUE
          S=0.7
          ITER=0
C
C         Processus iteratif:
C
10        CONTINUE
          ITER=ITER+1
C
C         Nombre maximal d'iterations atteint.
C
          IF(ITER.GT.ITMAX)THEN
            ERR=1
            GO TO 30
          ENDIF
C
```

```
C       Evaluation des asymptotes mobiles.
C
40      CALL LUVAR(L,U,XINF,XSUP,X(1+MOD(ITER-1,2)*NV),
        1          X(1+MOD(ITER,2)*NV),SIGNE,S,ITER,NV)
C
C       Evaluation des "move-limits".
C
        CALL AB(L,U,A,B,NV,X(1+MOD(ITER-1,2)*NV))
C
C       Verification du fait que le domaine admissible ne soit pas vide.
C
        CALL VIDE(NV,XINF,XSUP,A,B,X(1+MOD(ITER-1,2)*NV),MODIF)
        IF(MODIF.EQ..TRUE.)THEN
          WRITE(27,*)
          WRITE(27,*)'Impossibilite de construire un sous-probleme.'
          WRITE(27,*)'Modification du point de depart.'
          WRITE(27,*)
          GO TO 40
        ENDIF
C
C       Evaluation des fonction objectif et contraintes en le point de
C       linearisation.
C
        CALL EVAL(X(1+MOD(ITER-1,2)*NV),FCT,NV,DFJX,DFIJX,P,MAXCON)
C
C       Initialisations pour la routine E04KBF.
C
        IF(P.EQ.1)THEN
          ETA=0.0
        ELSE
          ETA=0.9
        ENDIF
        LIW=2
        LW=9*MAXCON
        CALL FUNCT(2,P,LAMBDA,F,G,IW,LIW,W,LW)
        N=P
        IPRINT=0
        LOCSCH=.TRUE.
        INTYPE=1
        MAXCAL=100*P
        XTOL=0.0
        STEPMX=1000000.0
        FEST=-1000000.0
        IBOUND=2
        LH=MAXCON*MAXCON/2
        DO 111 IND=1,LH
          HESL(IND)=0.
111     CONTINUE
        CALL HESSIEN(X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),L,U,
        1            D,XINF,XSUP,DFJX,DFIJX,LAMBDA,P,NV,HESD,
        2            A,B,MAXCON)
77      IFAIL=1
C
```

```
C         Resolution du sous-probleme.
C
          CALL E04KBF(N,FUNCT,MONIT,IPRINT,LOCSCH,INTYPE,E04LBS,
          1           MAXCAL,ETA,XTOL,STEPMX,FEST,IBOUND,BL,BU,
          2           LAMBDA,HESL,LH,HESD,ISTATE,F,G,IW,LIW,W,LW,IFAIL)
C
C         Corrections necessaires de la resolution.
C
          IF(IFAIL.EQ.2)THEN
            INTYPE=2
            GO TO 77
          ENDIF
          IF(IFAIL.EQ.4)THEN
            INTYPE=0
            GO TO 77
          ENDIF
C
C         Evaluation des variables primales et artificielles en la solution.
C
          CALL CALPRI(L,U,X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),
          1           XINF,XSUP,A,B,DFJX,DFIJX,LAMBDA,P,NV,Z,D,MAXCON)
C
C         Verification des conditions d'optimalite.
C
          IF(TSTOPT(X,Z,ITER,NV,P,LAMBDA,FCHAP,XINF,XSUP,EPS,MAXCON,
          1  MAXVAR).EQ..FALSE.)THEN
            CALL INTERM(ITER,X(1+MOD(ITER,2)*NV),Z,NV,P)
            GO TO 10
          ENDIF
          ERR=0
30        CONTINUE
          RETURN
          END
C
```

```
C     ************************************************************************
C
      SUBROUTINE AB(L,U,A,B,N,XK)
C     ===========================
C
C     Role : La routine calcule les valeurs des "move-limits" alpha et
C     ----   beta a l'iteration K.
C            Les autres parametres sont inchanges.
C            La routine fait appel a la routine ABJ.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration K (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration K (DP)
C       N : nombre de variables (INT)
C       XK : tableau de longueur egale au nombre de variables ,
C            contenant les composantes de l'itere Xk (DP)
C
C       Parametres de sortie:
C       --------------------
C       A : tableau de longueur egale au nombre de variables , contenant
C           les composantes de la "move-limit" alpha a l'iteration K (DP)
C       B : tableau de longueur egale au nombre de variables , contenant
C           les composantes de la "move-limit" beta a l'iteration K (DP)
C
C
C     Specifications:
C     --------------
C
      INTEGER N
      DOUBLE PRECISION L,U,A,B,XK
      DIMENSION L(N),U(N),A(N),B(N),XK(N)
C
C     Declarations:
C     ------------
C
      INTEGER J
C
C     Programme:
C     ---------
C
      DO 100 J=1,N
C
C       Mise a jour de la J-ieme composante des "move-limits".
C
        CALL ABJ(L(J),U(J),A(J),B(J),XK(J))
100   CONTINUE
      RETURN
      END
C
```

```
      SUBROUTINE ABJ(LJ,UJ,AJ,BJ,XKJ)
C     ===============================
C
C     Role:La routine calcule les valeurs des J-iemes composantes des
C     ---- "move-limits" alpha et beta a l'iteration K .
C          Les autres parametres sont inchanges.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration K (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration K (DP)
C       XKJ : J-ieme composante du point de linearisation a l'iteration
C             K (DP)
C
C       Parametres de sortie:
C       --------------------
C       AJ : J-ieme composante de la "move-limit" alpha a l'iteration
C            K (DP)
C       BJ : J-ieme composante de la "move-limit" beta a l'iteration
C            K (DP)
C
C
C     Specifications:
C     --------------
C
      DOUBLE PRECISION LJ,UJ,AJ,BJ,XKJ
C
C     Programme:
C     ---------
C
      AJ=0.9*LJ+0.1*XKJ
      BJ=0.9*UJ+0.1*XKJ
      RETURN
      END
C
```

```
C     ************************************************************
C
      SUBROUTINE CALPRI(L,U,XK,Y,XINF,XSUP,A,B,DFJX,DFIJX,
     1                  LAMBDA,P,N,Z,D,DIM)
C     ==================================================
C
C     Role : La routine calcule les variables primales et les variables
C     ----   artificielles en fonction des variables duales courantes.
C            Les parametres d'entree ne sont pas modifies .
C            La routine utilise la routine VARPRI.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration presente (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration presente (DP)
C      XK : tableau de longueur egale au nombre de variables ,
C            contenant les composantes du point de linearisation Xk (DP)
C       XINF : tableau de longueur egale au nombre de variables ,
C              contenant la borne inferieure sur X (DP)
C       XSUP : tableau de longueur egale au nombre de variables ,
C              contenant la borne superieure sur X (DP)
C       A : tableau de longueur egale au nombre de variables , contenant
C           la "move-limit" alpha a l'iteration presente (DP)
C       B : tableau de longueur egale au nombre de variables , contenant
C           la "move-limit" beta a l'iteration presente (DP)
C       DFJX : tableau de longueur egale au nombre de variables ,
C              contenant les derivees de la fonction objectif en le point
C              de linearisation Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales courantes (DP)
C       P : nombre de contraintes (INT)
C       N : nombre de variables (INT)
C       D : tableau de longueur egale au nombre de contraintes , contenant
C           les coefficients Di des variables artificielles de la fonction
C           objectif du probleme linearise (DP)
C       DIM : dimension maximale (INT)
C
C       Parametres de sortie:
C       --------------------
C       Y : tableau de longueur egale au nombre de variables , contenant
C           les variables primales calculees en fonction des variables
C           duales courantes (DP)
C       Z : tableau de longueur egale au nombre de contraintes , contenant
C           les variables artificielles calculees en fonction des variables
C           duales courantes (DP)
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER P,N,DIM
          DOUBLE PRECISION L,U,XK,Y,XINF,XSUP,A,B,DFJX,DFIJX,LAMBDA,Z,D
          DIMENSION L(N),U(N),XK(N),Y(N),XINF(N),XSUP(N),
         1          A(N),B(N),DFJX(N),DFIJX(DIM,N),LAMBDA(P),
         2          Z(P),D(P)
C
C         Declarations:
C         ------------
C
          INTEGER IND
C
C         Programme:
C         ---------
C
          DO 100 IND=1,N
C
C           Calcul de la IND-ieme variable primale.
C
            CALL VARPRI(L(IND),U(IND),XK(IND),Y(IND),XINF(IND),
         1              XSUP(IND),A(IND),B(IND),DFJX(IND),DFIJX,
         2              LAMBDA,IND,P,N,DIM)
  100     CONTINUE
C
C         Calcul des variables artificielles.
C
          DO 200 IND=1,P
            Z(IND)=-0.5+LAMBDA(IND)/(2*D(IND))
            IF(Z(IND).LT.0.)THEN
              Z(IND)=0.
            ENDIF
  200     CONTINUE
          RETURN
          END
C
```

```
C     ************************************************************************
C
      SUBROUTINE CONT(L,U,XK,Y,Z,FCT,DFJX,DFIJX,FCHAP,GRADLG,N,P,DIM)
C     ==============================================================
C
C     Role : calcule les valeurs des contraintes du sous-probleme
C     ----   primal linearise en XK , pour la valeur Y.  Cela revient
C            a calculer les derivees premieres de la fonction lagran-
C            gienne du probleme dual correspondant en Y.
C            Les parametres d'entree ne sont pas modifies.
C            Elle fait appel a la fonction LIN.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration presente (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration presente (DP)
C      XK : tableau de longueur egale au nombre de variables ,
C            contenant les composantes du point de linearisation Xk (DP)
C       Y : tableau de longueur egale au nombre de variables , contenant
C           les composantes du point primal correspondant aux variables
C           duales courantes (DP)
C       Z : tableau de longueur egale au nombre de contraintes ,
C           contenant les composantes des variables artificielles
C         correspondant aux variables duales courantes (DP)
C       FCT : tableau de longueur egale au nombre de contraintes + 1 ,
C             contenant l'evaluation de la fonction objectif et des
C             contraintes en le point de linearisation Xk (DP)
C       DFJX : tableau de longueur egale au nombre de variables ,
C              contenant les derivees de la fonction objectif en le point
C              de linearisation Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       FCHAP : tableau de longueur egale au nombre de contraintes ,
C               contenant les seconds membres des contraintes (DP)
C       N : nombre de variables (INT)
C       P : nombre de contraintes (INT)
C       DIM : dimension maximale (INT)
C
C       Parametres de sortie :
C       ---------------------
C       GRADLG : tableau de longueur egale au nombre de contraintes ,
C                contenant les derivees premieres de la fonction lagrangienne
C                du sous-probleme dual linearise en Xk , evaluees en Y (DP)
C
```

```
C
C       Specifications:
C       --------------
C
        INTEGER N,P,DIM
        DOUBLE PRECISION DFJX,DFIJX,XK,U,L,Y,Z,FCHAP,GRADLG,FCT
        DIMENSION DFJX(N),DFIJX(DIM,N),XK(N),U(N),L(N),Y(N),Z(P),
        1         FCHAP(P),GRADLG(P),FCT(P+1)
C
C       Declarations:
C       ------------
C
        INTEGER IND
        DOUBLE PRECISION LIN
C
C       Programme:
C       ---------
C
        DO 100 IND=1,P
C
C         La fonction LIN calcule la linearisee de la IND-ieme contrainte
C         primale en XK et evaluee en Y.
C
          GRADLG(IND)=-LIN(L,U,XK,Y,FCT,DFJX,DFIJX,N,P,IND,DIM)
        1             +Z(IND)+FCHAP(IND)
100     CONTINUE
        RETURN
        END
C
```

```
C     ************************************************************************
C
      SUBROUTINE FUNCT(IFLAG,N,XC,FC,GC,IW,LIW,W,LW)
C     ==============================================
C
C     Role : Cette routine est demandee par la routine E04KBF et evalue
C     ----   la fonction duale et (si IFLAG=2) les derivees premieres de
C            cette fonction en les variables duales courantes.  Il est
C            a noter que ces dernieres sont rendues positives en mettant
C            a 0 les composantes strictement negatives.
C            Cette routine utilise les routines et fonctions ACCEPT ,
C            CALPRI , CONT , LIN.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       IFLAG : indique s'il faut calculer uniquement la fonction duale
C               (1) ou la fonction duale et ses derivees premieres (2).
C               Les evaluations se font en le point courant XC (INT)
C       N : nombre de composantes de XC.  Ici , cela vaut le nombre de
C           contraintes (INT)
C       XC : tableau de longueur egale au nombre de contraintes ,
C            contenant les variables duales courantes (DP)
C
C       Note : IW , LIW , W , LW : ce sont des variables de travail qui
C              ne sont pas modifies dans FUNCT.
C              IW est un tableau de longueur LIW >= 2 (INT)
C              W est un tableau de longueur LW >= 3*N (DP)
C
C       Commons:
C       -------
C       INT : ITER : numero de l'iteration (INT)
C             NV : nombre de variables primales (INT)
C             P : nombre de contraintes (INT)
C       AUX : L : tableau contenant les composantes de l'asymptote L
C                 a l'iteration presente (DP)
C             U : tableau contenant les composantes de l'asymptote U
C                 a l'iteration presente (DP)
C             A : tableau contenant les composantes de la "move-limit"
C                 alpha a l'iteration presente (DP)
C             B : tableau contenant les composantes de la "move-limit"
C                 beta a l'iteration presente (DP)
C             FCT : tableau contenant la fonction objectif et les
C                   contraintes evaluees en le point de linearisation Xk
C                   (DP)
C             DFJX : tableau contenant les derivees de la fonction
C                    objectif evaluees en Xk , point de linearisation (DP)
C             DFIJX : tableau dont la I-ieme ligne contient les derivees
C                     de la I-ieme contrainte evaluees en le point de
C                     linearisation Xk (DP)
C             D : tableau contenant les coefficients Di des variables
C                 artificielles dans la fonction objectif du probleme
C                 linearise (DP)
C
```

```
C       Programme:
C       ---------
C
C       Verification de l'admissibilite des variables duales courantes.
C
        CALL ACCEPT(XC,P)
C
C       Calcul des variables primales et artificielles en fonction des
C       variables duales courantes.
C
        CALL CALPRI(L,U,X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),
       1            XINF,XSUP,A,B,DFJX,DFIJX,XC,P,NV,Z,D,MAXCON)
        IF(IFLAG.EQ.2)THEN
C
C         Evaluation des contraintes primales.
C
          CALL CONT(L,U,X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),
       1            Z,FCT,DFJX,DFIJX,FCHAP,GC,NV,P,MAXCON)
          AUX1=0.
          DO 100 IND=1,P
            AUX1=AUX1+XC(IND)*GC(IND)
100       CONTINUE
        ELSE
          CALL CONT(L,U,X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),
       1            Z,FCT,DFJX,DFIJX,FCHAP,GRADLG,NV,P,MAXCON)
          AUX1=0.
          DO 200 IND=1,P
            AUX1=AUX1+XC(IND)*GRADLG(IND)
200       CONTINUE
        ENDIF
        AUX2=0.
        DO 300 IND=1,P
          AUX2=AUX2-D(IND)*(Z(IND)+Z(IND)**2)
300     CONTINUE
C
C       Evaluation de la fonction lagrangienne.
C
        FC=-LIN(L,U,X(1+MOD(ITER-1,2)*NV),X(1+MOD(ITER,2)*NV),FCT,
       1  DFJX,DFIJX,NV,P,0,MAXCON)+AUX1+AUX2
        RETURN
        END
C
```

```
C     ************************************************************************
C
      SUBROUTINE HESSIEN(X,Y,L,U,D,XINF,XSUP,DFJX,DFIJX,LAMBDA,
     1                   P,N,HES,A,B,DIM)
C     ========================================================
C
C     Role : Cette routine calcule la diagonale du hessien de la fonction
C     ----   lagrangienne du probleme linearise en X et evalue en Y.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C     X : tableau de longueur egale au nombre de variables ,
C           contenant les composantes du point de linearisation Xk (DP)
C       Y : tableau de longueur egale au nombre de variables , contenant
C           les variables primales correspondant aux variables duales
C           courantes (DP)
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration presente (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration presente (DP)
C       D : tableau de longueur egale au nombre de contraintes , contenant
C           les coefficients Di des variables artificielles de la fonction
C           objectif du probleme linearise (DP)
C       XINF : tableau de longueur egale au nombre de variables ,
C              contenant la borne inferieure sur X (DP)
C      XSUP : tableau de longueur egale au nombre de variables ,
C              contenant la borne superieure sur X (DP)
C       DFJX : tableau de longueur egale au nombre de variables ,
C              contenant les derivees de la fonction objectif en le point
C              de linearisation Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales courantes (DP)
C       P : nombre de contraintes (INT)
C       N : nombre de variables (INT)
C       A : tableau de longueur egale au nombre de variables , contenant
C           la "move-limit" alpha a l'iteration presente (DP)
C       B : tableau de longueur egale au nombre de variables , contenant
C           la "move-limit" beta a l'iteration presente (DP)
C       DIM : dimension maximale (INT)
C
C       Parametres de sortie:
C       --------------------
C       HES : tableau de longueur egale au nombre de contraintes , contenant
C             le hessien de la fonction lagrangienne evalue en le point
C             courant.  Plus precisement , HES ne contient que la diagonale
C             de ce hessien (DP)
C
```

```
C
C       Specifications:
C       ---------------
C
        INTEGER P,N,DIM
        DOUBLE PRECISION X,Y,L,U,DFJX,DFIJX,LAMBDA,HES,
     1    D,XINF,XSUP,A,B
        DIMENSION X(N),Y(N),L(N),U(N),DFJX(N),DFIJX(DIM,N),LAMBDA(P),
     1          HES(DIM),D(P),XINF(N),XSUP(N),A(N),B(N)
C
C       Declarations:
C       -------------
C
        INTEGER I,K,IND
        DOUBLE PRECISION AUX1,AUX2,AUX3,AUX4
C
C       Programme:
C       ----------
C
        DO 100 I=1,P
            HES(I)=1/(2*D(I))
            DO 300 K=1,N
              IF(Y(K).EQ.MAX(A(K),XINF(K)).OR.Y(K).EQ.MIN(B(K),XSUP(K)))
     1          GO TO 300
              IF(DFIJX(I,K).GE.0)THEN
                AUX1=(((U(K)-X(K))**2)/((U(K)-Y(K))**2))*DFIJX(I,K)
              ELSE
                AUX1=(((X(K)-L(K))**2)/((Y(K)-L(K))**2))*DFIJX(I,K)
              ENDIF
              AUX3=0
              AUX4=0
              DO 400 IND=1,P
                IF(DFIJX(IND,K).GE.0)THEN
                  AUX3=AUX3+LAMBDA(IND)*DFIJX(IND,K)
                ELSE
                  AUX4=AUX4-LAMBDA(IND)*DFIJX(IND,K)
                ENDIF
400           CONTINUE
              IF(DFJX(K).GE.0)THEN
                IF(DFIJX(I,K).GE.0)THEN
                  AUX2=(U(K)-X(K))*(X(K)-L(K))*0.5*(L(K)-U(K))*
     1                ((DFJX(K)+AUX3)**(-0.5))*(AUX4**0.5)*DFIJX(I,K)
                ELSE
                  AUX2=(U(K)-X(K))*(X(K)-L(K))*0.5*(L(K)-U(K))*
     1                ((DFJX(K)+AUX3)**0.5)*(AUX4**(-0.5))*DFIJX(I,K)
                ENDIF
                AUX2=AUX2/(((U(K)-X(K))*((DFJX(K)+AUX3)**0.5)+
     1              (X(K)-L(K))*(AUX4**0.5))**2)
              ELSE
                IF(DFIJX(I,K).GE.0)THEN
                  AUX2=(L(K)-U(K))*0.5*(X(K)-L(K))*(U(K)-X(K))*
     1                (AUX3**(-0.5))*((-DFJX(K)+AUX4)**0.5)*DFIJX(I,K)
                ELSE
                  AUX2=(L(K)-U(K))*0.5*(U(K)-X(K))*(X(K)-L(K))*
     1                (AUX3**0.5)*((-DFJX(K)+AUX4)**(-0.5))*DFIJX(I,K)
                ENDIF
                AUX2=AUX2/(((X(K)-L(K))*((-DFJX(K)+AUX4)**0.5)+
     1              (U(K)-X(K))*(AUX3**0.5))**2)
              ENDIF
              HES(I)=HES(I)-AUX1*AUX2
300         CONTINUE
100     CONTINUE
        RETURN
        END
C
```

```
C     ************************************************************************
C
      DOUBLE PRECISION FUNCTION LIN(L,U,XK,Y,FCT,DFJX,DFIJX,N,P,I,DIM)
C     ================================================================
C
C     Role : contient la linearisee de la I-ieme contrainte primale en
C     ----   XK et evaluee en Y.  Si I vaut 0 , on calcule la linearisee
C            de la fonction objectif.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration presente (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration presente (DP)
C      XK : tableau de longueur egale au nombre de variables ,
C            contenant les composantes du point de linearisation Xk (DP)
C       Y : tableau de longueur egale au nombre de variables , contenant
C           les composantes du point primal correspondant aux variables
C           duales courantes (DP)
C       FCT : tableau de longueur egale au nombre de contraintes + 1 ,
C             contenant l'evaluation de la fonction objectif et des
C             contraintes en le point de linearisation Xk (DP)
C       DFJX : tableau de longueur egale au nombre de variables ,
C              contenant les derivees de la fonction objectif en le point
C              de linearisation Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       N : nombre de variables (INT)
C       P : nombre de contraintes (INT)
C       I : numero de la contrainte que l'on veut lineariser (I=0 => il
C           s'agit de la fonction objectif) (INT)
C       DIM : dimension maximale (INT)
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER I,N,P,DIM
          DOUBLE PRECISION DFJX,DFIJX,U,L,Y,XK,FCT
          DIMENSION DFJX(N),DFIJX(DIM,N),U(N),L(N),Y(N),XK(N),FCT(P+1)
C
C         Declarations:
C         ------------
C
          INTEGER J
          DOUBLE PRECISION AUX
C
C         Programme:
C         ---------
C
          LIN=FCT(I+1)
          DO 100 J=1,N
            IF(I.EQ.0)THEN
              AUX=DFJX(J)
            ELSE
              AUX=DFIJX(I,J)
            ENDIF
            IF(AUX.GE.0)THEN
              LIN=LIN+(((U(J)-XK(J))**2)/(U(J)-Y(J))-(U(J)-XK(J)))
     1             *AUX
            ELSE
              LIN=LIN+((XK(J)-L(J))-((XK(J)-L(J))**2)/(Y(J)-L(J)))
     1             *AUX
            ENDIF
100       CONTINUE
          RETURN
          END
C
```

```
C     ************************************************************************
C
      SUBROUTINE LUVAR(L,U,XINF,XSUP,XK,XPREC,SIGNE,S,K,N)
      ====================================================
C
C
C     Role : La routine calcule les asymptotes mobiles L et U a l'iteration
C     ----   K.  Elle met a jour le parametre SIGNE contenant (Xk)-(Xk-1)
C            si K>=2.
C            Les autres parametres sont inchanges.
C            La routine fait appel a la routine LUVARJ.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       L : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote L a l'iteration K-1 (DP)
C       U : tableau de longueur egale au nombre de variables , contenant
C           les composantes de l'asymptote U a l'iteration K-1 (DP)
C       XINF : tableau de longueur egale au nombre de variables ,
C              contenant les composantes de la borne inferieure sur X (DP)
C       XSUP : tableau de longueur egale au nombre de variables ,
C              contenant les composantes de la borne superieure sur X (DP)
C       XK : tableau de longueur egale au nombre de variables , contenant
C            les composantes du point de linearisation Xk (DP)
C       XPREC : tableau de longueur egale au nombre de variables ,
C               contenant les composantes de l'itere precedent Xk-1 (DP)
C       SIGNE : tableau de longueur egale au nombre de variables ,
C               contenant la difference entre les iteres Xk-1 et Xk-2 (DP)
C       S : reel < 1 (DP)
C       K : iteration (INT)
C       N : nombre de variables (INT)
C
C       Note : Si K=1 , les parametres L,U,XPREC,SIGNE sont indefinis.
C              Si K=2 , le parametre SIGNE est indefini .
C
C       Parametres de sortie:
C       --------------------
C       L : contient les composantes de l'asymptote L a l'iteration K (DP)
C       U : contient les composantes de l'asymptote U a l'iteration K (DP)
C       SIGNE : (seulement si K >= 2) difference entre les iteres Xk et
C               Xk-1 (DP)
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER K,N
          DOUBLE PRECISION L,U,XINF,XSUP,SIGNE,S,XK,XPREC
          DIMENSION L(N),U(N),XINF(N),XSUP(N),SIGNE(N),XK(N),
          1         XPREC(N)
C
C         Declarations:
C         ------------
C
          INTEGER J
C
C         Programme:
C         ---------
C
          DO 100 J=1,N
C
C           Mise a jour de la J-ieme composante des asymptotes mobiles.
C
            CALL LUVARJ(L(J),U(J),XINF(J),XSUP(J),XK(J),K,XPREC(J),
          1             SIGNE(J),S)
100       CONTINUE
          RETURN
          END
C
```

```
      SUBROUTINE LUVARJ(LJ,UJ,XJI,XJS,XKJ,K,XJPREC,SIGNE,S)
C     ====================================================
C
C     Role:La routine calcule les J-iemes composantes des nouvelles
C     ---- asymptotes mobiles L et U a l'iteration K.  Elle met a jour le
C          parametre SIGNE contenant Xk-1(J)-Xk(J) si K >= 2.
C          Les autres parametres sont inchanges.
C
C     Documentation:
C     --------------
C       Parametres d'entree:
C       --------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration K-1 (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration K-1 (DP)
C       XJI : J-ieme composante de la borne inferieure sur X (DP)
C       XJS : J-ieme composante de la borne superieure sur X (DP)
C       XKJ : J-ieme composante du point de linearisation a l'iteration
C             K (DP)
C       K : iteration (INT)
C       XJPREC : J-ieme composante du point de linearisation a l'iteration
C                K-1 (DP)
C       SIGNE : difference entre la J-ieme composante de Xk-1 et la J-ieme
C               composante de Xk-2 (DP)
C       S : reel < 1 (DP)
C
C       Note: Si K=1 , les parametres LJ,UJ,XJPREC,SIGNE sont indefinis.
C             Si K=2 , le parametre SIGNE est indefini.
C
C       Parametres de sortie:
C       ---------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration K (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration K (DP)
C       SIGNE : (seulement si K >=2) difference entre les J-iemes composantes
C               de Xk-1 et Xk
C
C
C     Specifications:
C     ---------------
C
      DOUBLE PRECISION LJ,UJ,XJI,XJS,XKJ,XJPREC,SIGNE,S
      INTEGER K
C
C     Programme:
C     ----------
C
      IF((K.EQ.1).OR.(K.EQ.2))THEN
        LJ=XKJ-(XJS-XJI)
        UJ=XKJ+(XJS-XJI)
      ELSEIF((SIGNE*(XKJ-XJPREC)).GE.0)THEN
        LJ=XKJ-(XJPREC-LJ)/S
        UJ=XKJ+(UJ-XJPREC)/S
      ELSE
        LJ=XKJ-S*(XJPREC-LJ)
        UJ=XKJ+S*(UJ-XJPREC)
      ENDIF
      IF(K.GE.2)THEN
        SIGNE=(XKJ-XJPREC)
      ENDIF
      RETURN
      END
```

```
C     ******************************************************************************
C
      DOUBLE PRECISION FUNCTION MINDJ(LJ,UJ,XKJ,DFJXJ,DFIJX,
     1                                LAMBDA,J,P,N,ERR,DIM)
C     ====================================================
C
C     Role : La fonction calcule la variable primale d'indice J en fonction
C    ----    des variables duales courantes et sans tenir compte des bornes.
C            Le parametre ERR permet de signaler si le calcul est
C            impossible , ce qui indiquerait que n'importe quelle valeur
C            est acceptable .
C              Les parametres d'entree ne sont pas modifies.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration presente (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration presente (DP)
C       XKJ : J-ieme composante du point de linearisation Xk (DP)
C       DFJXJ : derivee par rapport a la J-ieme variable de la fonction
C               objectif et evaluee en Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales courantes (DP)
C       J : indice de la variable primale que l'on va calculer (INT)
C       P : nombre de contraintes (INT)
C       N : nombre de variables (INT)
C       DIM : dimension maximale (INT)
C
C       Parametres de sortie:
C       --------------------
C       ERR : indique s'il a ete possible oui ou non (0 ou 1) de calculer
C             la variable primale.  Si c'est non , cela indique que toute
C             valeur pour cette variable minimise le lagrangien.
C
```

```
C
C         Specifications:
C         ---------------
C
          INTEGER J,P,ERR,N,DIM
          DOUBLE PRECISION LJ,UJ,XKJ,DFJXJ,LAMBDA,DFIJX
          DIMENSION LAMBDA(P),DFIJX(DIM,N)
C
C         Declarations:
C         -------------
C
          INTEGER IND
          DOUBLE PRECISION AUX1,AUX2
C
C         Programme:
C         ----------
C
          ERR=0
          AUX1=0.
          AUX2=0.
          DO 100 IND=1,P
            IF(DFIJX(IND,J).GE.0)THEN
              AUX1=AUX1+DFIJX(IND,J)*LAMBDA(IND)
            ELSE
              AUX2=AUX2-DFIJX(IND,J)*LAMBDA(IND)
            ENDIF
100       CONTINUE
          IF(DFJXJ.GE.0)THEN
            AUX1=AUX1+DFJXJ
            IF(AUX1.EQ.0.AND.AUX2.EQ.0)THEN
              ERR=1
              RETURN
            ELSE
              MINDJ=(LJ*(UJ-XKJ)*(AUX1**0.5)+UJ*(XKJ-LJ)*(AUX2**0.5))/
         1          ((UJ-XKJ)*(AUX1**0.5)+(XKJ-LJ)*(AUX2**0.5))
            ENDIF
          ELSE
            AUX2=AUX2-DFJXJ
            IF(AUX1.EQ.0.AND.AUX2.EQ.0)THEN
              ERR=1
              RETURN
            ELSE
              MINDJ=(UJ*(XKJ-LJ)*(AUX2**0.5)+LJ*(UJ-XKJ)*(AUX1**0.5))/
         1          ((XKJ-LJ)*(AUX2**0.5)+(UJ-XKJ)*(AUX1**0.5))
            ENDIF
          ENDIF
          RETURN
          END
C
```

```
C     ************************************************************************
C
      DOUBLE PRECISION FUNCTION MINDVA(LJ,UJ,XKJ,YJ,DFJXJ,DFIJX,
     1                                 LAMBDA,J,N,P,DIM)
C     ========================================================
C
C     Role : La fonction evalue en un point YJ , la derivee par rapport
C     ----   a la J-ieme variable , de la partie du lagrangien qu'il
C            faut minimiser pour pouvoir calculer la J-ieme variable
C            primale en fonction des variables duales courantes .
C            Les parametres d'entree ne sont pas modifies .
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration presente (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration presente (DP)
C       XKJ : J-ieme composante du point de linearisation Xk (DP)
C       YJ : endroit ou l'on veut evaluer la fonction (DP)
C       DFJXJ : derivee par rapport a la J-ieme variable de la fonction
C               objectif evaluee en Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables) , dont
C               la I-ieme ligne contient les derivees de la I-ieme contrainte
C               evaluees en Xk (DP)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales courantes (DP)
C       J : indique la variable que l'on va traiter (INT)
C       N : nombre de variables (INT)
C       P : nombre de contraintes (INT)
C       DIM : dimension maximale (INT)
C
```

```
C
C         Specifications:
C         ---------------
C
          INTEGER N,P,J,DIM
          DOUBLE PRECISION DFJXJ,UJ,XKJ,DFIJX,LAMBDA,LJ,YJ
          DIMENSION DFIJX(DIM,N),LAMBDA(P)
C
C         Declarations:
C         -------------
C
          INTEGER K
          DOUBLE PRECISION MINLAG
C
C         Programme:
C         ----------
C
            IF(DFJXJ.GE.0)THEN
              MINLAG=(((UJ-XKJ)**2)/((UJ-YJ)**2))*DFJXJ
            ELSE
              MINLAG=(((XKJ-LJ)**2)/((YJ-LJ)**2))*DFJXJ
            ENDIF
            DO 200 K=1,P
              IF(DFIJX(K,J).GE.0)THEN
                MINLAG=MINLAG+(((UJ-XKJ)**2)/
     1                ((UJ-YJ)**2))*LAMBDA(K)*DFIJX(K,J)
              ELSE
                MINLAG=MINLAG+(((XKJ-LJ)**2)/
     1                ((YJ-LJ)**2))*LAMBDA(K)*DFIJX(K,J)
              ENDIF
200           CONTINUE
          MINDVA=MINLAG
          RETURN
          END
C
```

```
C
      SUBROUTINE MONIT(N,XC,FC,GC,ISTATE,GPJNRM,COND,POSDEF,
     1                 NITER,NF,IW,LIW,W,LW)
C     ====================================================
C
C     Role:routine demandee par E04KBF faisant diverses impressions.
C     ---- Dans le cas present , il n'y a aura pas d'impressions.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       N : nombre de composantes de XC , ici cela vaut le nombre de
C           contraintes (INT)
C       XC : tableau de longueur N contenant les composantes du point
C            courant X (ici les variables duales courantes) (DP)
C       FC : valeur de la fonction lagrangienne au point XC (DP)
C       GC : tableau de longueur N contenant les derivees de la fonction
C            lagrangienne evaluees en XC (DP)
C       ISTATE : tableau de longueur N indiquant si les variables courantes
C                XC sont fixees a leurs bornes ou sont libres (INT)
C       GPJNRM : norme euclidienne du gradient projete (DP)
C       COND : rapport du plus grand element sur le plus petit de la matrice
C              D (DP)
C       POSDEF : .TRUE. (LOG)
C       NITER : nombre d'iterations effectuees jusqu'a present (INT)
C       NF : nombre d'appel de la routine FUNCT (INT)
C
C       Note : IW , LIW , W , LW sont des variables de travail
C
C
C     Specifications:
C     --------------
C
      LOGICAL POSDEF
      INTEGER N,ISTATE(N),NITER,NF,LIW,IW(LIW),LW
      DOUBLE PRECISION XC(N),FC,GC(N),GPJNRM,COND,W(LW)
C
C     Programme:
C     ---------
C
      RETURN
      END
C
```

```
C     ************************************************************************
C
      LOGICAL FUNCTION TSTOPT(X,Z,ITER,N,P,LAMBDA,FCHAP,XINF,XSUP,
     1                        EPS)
C     ============================================================
C
C     Role : La fonction indique si oui ou non , on se trouve a l'optimum
C     ----   du probleme , en fonction de la precision souhaitee EPS.
C            La fonction vaut .TRUE. si c'est le cas et .FALSE. sinon.
C            Elle fait appel a la routine EVAL .
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       X : tableau de longueur egale a 2 fois le nombre de variables ,
C           contenant le point de linearisation Xk et le point qui est
C           solution optimale du sous-probleme linearise (DP)
C       Z : tableau delongueur egale au nombre de contraintes , contenant
C           les variables artificielles , solutions optimales du
C           sous-probleme linearise (DP)
C       ITER : numero de l'iteration (INT)
C       N : nombre de variables (INT)
C       P : nombre de contraintes (INT)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales a l'optimum du probleme
C                linearise (DP)
C       FCHAP : tableau delongueur egale au nombre de contraintes ,
C               contenant les seconds membres des contraintes (DP)
C       EPS : precision souhaitee (DP)
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER ITER,N,P,MAXCON,MAXVAR
          PARAMETER(MAXCON=100)
          PARAMETER(MAXVAR=100)
          DOUBLE PRECISION X,LAMBDA,FCHAP,EPS,Z,XINF,XSUP
          DIMENSION X(2*MAXVAR),LAMBDA(P),FCHAP(P),Z(P),XINF(N),XSUP(N)
C
C         Declarations:
C         ------------
C
          INTEGER IND,IND1
          DOUBLE PRECISION AUX,FCT,DFJX,DFIJX
          DIMENSION FCT(MAXCON+1),DFJX(MAXVAR),DFIJX(MAXCON,MAXVAR)
C
C         Programme:
C         ---------
C
          TSTOPT=.TRUE.
C
C         Evaluation des fonction objectif et contraintes en le point
C         traite.
C
          CALL EVAL(X(1+MOD(ITER,2)*N),FCT,N,DFJX,DFIJX,P,MAXCON)
C
C         Verification des conditions de Kuhn-Tucker.
C
          DO 100 IND=1,N
            IF(X(1+MOD(ITER,2)*N+IND-1).EQ.XINF(IND).OR.
         1    X(1+MOD(ITER,2)*N+IND-1).EQ.XSUP(IND))GO TO 100
            AUX=DFJX(IND)
            DO 200 IND1=1,P
              AUX=AUX+LAMBDA(IND1)*DFIJX(IND1,IND)
200         CONTINUE
            IF(ABS(AUX).GT.EPS)THEN
              TSTOPT=.FALSE.
              RETURN
            ENDIF
100       CONTINUE
          DO 300 IND=1,P
            AUX=LAMBDA(IND)*(FCT(IND+1)-FCHAP(IND))
            IF(ABS(AUX).GT.EPS)THEN
              TSTOPT=.FALSE.
              RETURN
            ENDIF
300       CONTINUE
C
C         Verification de l'admissibilite du point traite.
C
          DO 400 IND=1,P
            IF(FCT(IND+1)-FCHAP(IND).GT.EPS)THEN
              TSTOPT=.FALSE.
            ENDIF
400       CONTINUE
          RETURN
          END
C
```

```
**********************************
C
      SUBROUTINE VARPRI(LJ,UJ,XKJ,YJ,XJI,XJS,AJ,BJ,DFJXJ,DFIJX,
     1                  LAMBDA,J,P,N,DIM)
C     ========================================================
C
C
C     Role : La routine calcule la J-ieme composante de la variable
C     ----   primale en fonction des variables duales courantes et des
C            bornes.
C            Les parametres d'entree ne sont pas modifies.
C            La routine utilise les fonctions MINDJ et MINDVA .
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       LJ : J-ieme composante de l'asymptote L a l'iteration presente (DP)
C       UJ : J-ieme composante de l'asymptote U a l'iteration presente (DP)
C       XKJ : J-ieme composante du point de linearisation Xk (DP)
C       XJI : J-ieme composante de la borne inferieure sur X (DP)
C       XJS : J-ieme composante de la borne superieure sur X (DP)
C       AJ : J-ieme composante de la "move-limit" alpha a l'iteration
C            presente (DP)
C       BJ : J-ieme composante de la "move-limit" beta a l'iteration
C            presente (DP)
C       DFJXJ : derivee par rapport a la J-ieme variable de la fonction
C               objectif et evaluee en Xk (DP)
C       DFIJX : tableau (nombre de contraintes x nombre de variables ) ,
C               dont la I-ieme ligne contient les derivees de la I-ieme
C               contrainte , evaluees en le point de linearisation Xk (DP)
C       LAMBDA : tableau de longueur egale au nombre de contraintes ,
C                contenant les variables duales courantes (DP)
C       J : indice de la variable primale que l'on va calculer (INT)
C       P : nombre de contraintes (INT)
C       N : nombre de variables (INT)
C       DIM : dimension maximale (INT)
C
C       Parametres de sortie:
C       --------------------
C       YJ : J-ieme composante de la variable primale.  Cette composante
C            a ete calculee en fonction des variables duales et en tenant
C            compte des bornes et des particularites de la fonction
C            lagrangienne.
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER J,P,N,DIM
          DOUBLE PRECISION DFJXJ,LJ,UJ,XKJ,LAMBDA,DFIJX,AJ,BJ,
         1                 XJI,XJS,YJ
          DIMENSION LAMBDA(P),DFIJX(DIM,N)
C
C         Declarations:
C         ------------
C
          INTEGER ERR
          DOUBLE PRECISION AUX,MINDJ,MINDVA
C
C         Programme:
C         ---------
C
C         La fonction MINDJ calcule la J-ieme variable primale sans
C         tenir compte des bornes.  La variable ERR a la valeur 1 si ce
C         calcul est impossible.
C
          AUX=MINDJ(LJ,UJ,XKJ,DFJXJ,DFIJX,LAMBDA,J,P,N,ERR,DIM)
          IF(ERR.NE.1)THEN
C
C           La fonction MINDVA evalue la derivee premiere d'une partie
C           de la fonction lagrangienne evaluee en MAX(AJ,XJI) , puis
C           en MIN(BJ,XJS).
C
            IF(MINDVA(LJ,UJ,XKJ,MAX(AJ,XJI),DFJXJ,DFIJX,LAMBDA,
         1            J,N,P,DIM).GE.0)THEN
              YJ=MAX(AJ,XJI)
            ELSEIF(MINDVA(LJ,UJ,XKJ,MIN(BJ,XJS),DFJXJ,DFIJX,LAMBDA,
         1                J,N,P,DIM).LE.0)THEN
              YJ=MIN(BJ,XJS)
            ELSE
              YJ=AUX
            ENDIF
          ELSE
            YJ=MAX(AJ,XJI)
          ENDIF
          RETURN
          END
C
```

```
C     ************************************************************************
C
      SUBROUTINE VIDE(NV,XINF,XSUP,A,B,X,MODIF)
C     =========================================
C
C     Role:verifie si l'intervalle defini par les asymptotes mobiles a une
C     ---- intersection avec celui defini par les bornes.  Si ce n'est pas
C          le cas , le point de linearisation est ajuste de sorte que cela
C          le devienne.  La variable MODIF a la valeur .TRUE. si un tel
C          ajustement s'est avere necessaire.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       NV : nombre de variables (INT)
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X courantes (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X courantes (DP)
C       A : tableau de longueur egale au nombre de variables , contenant
C           les composantes de la "move-limit" alpha calculee (DP)
C       B : tableau de longueur egale au nombre de variables , contenant
C           les composantes de la "move-limit" beta calculee (DP)
C       X : tableau de longueur egale au nombre de variables , contenant
C           le point de linearisation courant (DP)
C
C       Parametres de sortie:
C       --------------------
C       X : tableau de longueur egale au nombre de variables , contenant
C           les composantes du point de linearisation eventuellement
C           modifies (DP)
C       MODIF : indique si une modification du point de linearisation a ete
C               effectuee.  Dans ce cas , MODIF a la valeur .TRUE. , sinon
C               elle a la valeur .FALSE.
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER NV
          DOUBLE PRECISION XINF,XSUP,A,B,X
          LOGICAL MODIF
          DIMENSION XINF(NV),XSUP(NV),A(NV),B(NV),X(NV)
C
C         Declarations:
C         ------------
C
          INTEGER IND
C
C         Programme:
C         ---------
C
          MODIF=.FALSE.
          DO 100 IND=1,NV
            IF(A(IND).GE.XSUP(IND))THEN
              X(IND)=(XSUP(IND)-XINF(IND))/2.+XINF(IND)
              MODIF=.TRUE.
            ENDIF
            IF(B(IND).LE.XINF(IND))THEN
              X(IND)=(XSUP(IND)-XINF(IND))/2.+XINF(IND)
              MODIF=.TRUE.
            ENDIF
100       CONTINUE
          RETURN
          END
C
```

2e Partie :

- Routines de contrôle.

- Interpréteur.

```
C     ************************************************************************
C
      SUBROUTINE OPTSUP(EPS,BINF,BSUP,ERR,IT,ITTOT)
C     =============================================
C
C     Role:routine controlant l'optimisation i.e. s'occupant de la gestion
C     ---- des bornes fictives , du probleme de cyclage.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       EPS : precision souhaitee (DP)
C       BINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X donnees par l'utilisateur (DP)
C       BSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X donnees par l'utilisateur (DP)
C       ITTOT : nombre maximal d'iterations permises pour le processus
C               general de minimisation (DP)
C
C       Parametres de sortie:
C       --------------------
C       ERR : indique si le processus s'est deroule avec succes (ERR=0) ,
C             si le nombre maximal total d'iterations a ete atteint (ERR=2) ,
C             ou encore si le nombre maximal d'iterations maximal a ete
C             atteint lors d'une minimisation et qu'il n'est pas possible
C             de modifier un parametre en vue de traiter une autre
C             minimisation (INT)
C       IT : nombre d'iterations total effectue (INT)
C
C       Commons:
C       -------
C       INT : ITER : numero de l'iteration (INT)
C             NV : nombre de variables primales (INT)
C             P : nombre de contraintes (INT)
C       DON1 : XINF : tableau contenant la borne inferieure sur X (DP)
C            XSUP : tableau contenant la borne superieure sur X (DP)
C       DON2 : X : tableau contenant le point de linearisation Xk et
C                  contenant , a l'entree , les variables primales
C                  evaluees en fonction des anciennes variables duales
C                  et a la sortie , les variables primales evaluees en
C                  fonction des nouvelles variables duales (DP)
C              FCHAP : tableau contenant les seconds membres des
C                      contraintes (DP)
C              Z : tableau contenant , a l'entree , les variables
C                  artificielles evaluees en fonction des anciennes
C                variables duales et a la sortie , les variables
C                  artificielles evaluees en fonction des nouvelles
C                  variables duales (DP)
C
```

```
C
C       Specifications:
C       ---------------
C
        INTEGER ERR,NV,ITTOT
        DOUBLE PRECISION BINF,BSUP,EPS
        DIMENSION BINF(NV),BSUP(NV)
C
C       Declarations:
C       -------------
C
        INTEGER IND,ITMAX,P,ITER,MAXVAR,MAXCON,IT
        PARAMETER(MAXVAR=100)
        PARAMETER(MAXCON=100)
        DOUBLE PRECISION XINF,XSUP,X,FCHAP,Z
        LOGICAL MODIF
        COMMON/INT/ITER,NV,P
        COMMON/DON1/XINF(MAXVAR),XSUP(MAXVAR)
        COMMON/DON2/X(2*MAXVAR),FCHAP(MAXCON),Z(MAXCON)
C
C       Programme:
C       ----------
C
C       Initialisation des bornes inexistantes.
C
        CALL INITBO(NV,X,BINF,BSUP,XINF,XSUP)
        ITMAX=100
        IT=0
C
C       Optimisation.
C
10      IF(IT.GT.ITTOT)THEN
          ERR=2
          RETURN
        ENDIF
        CALL OPT(EPS,ERR,ITMAX)
        IF(ERR.EQ.1)THEN
C
C         Cas presentant les caracteristiques d'un cyclage du a un mauvais
C         choix des bornes fictives.  Ajustement de celles-ci.
C
          CALL CYCLE(NV,BINF,BSUP,XINF,XSUP,MODIF)
          IF(MODIF.EQ..TRUE.)THEN
            WRITE(27,*)
            WRITE(27,*)'AJUSTEMENT DES BORNES FICTIVES '
            WRITE(27,*)'CAUSE:divergence.'
            WRITE(27,*)
            IT=IT+ITER
            GO TO 10
          ENDIF
        ELSE
C
C         Cas ou les bornes fictives sont actives.  Ajustement de celles-ci.
C
          CALL CORRBO(EPS,NV,ITER,XINF,XSUP,BINF,BSUP,X,MODIF)
          IF(MODIF.EQ..TRUE.)THEN
            WRITE(27,*)
            WRITE(27,*)'AJUSTEMENT DES BORNES FICTIVES'
            WRITE(27,*)'CAUSE:bornes actives'
            WRITE(27,*)
            IT=IT+ITER
            GO TO 10
          ENDIF
        ENDIF
        RETURN
        END
C
```

```
C     ************************************************************************
C
      SUBROUTINE CORRBO(EPS,NV,ITER,XINF,XSUP,BINF,BSUP,X,MODIF)
C     ==========================================================
C
C     Role:lorsqu'une borne fictive devient active , celle-ci est ajustee
C     ---- (On elargit le domaine en reculant la borne fictive active.)
C          La variable MODIF est mise a .TRUE. si un ajustement s'est
C          avere necessaire.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       EPS : precision souhaitee (DP)
C       NV : nombre de variables (INT)
C       ITER : numero de l'iteration (INT)
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X courantes (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X courantes (DP)
C       BINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X donnees par l'utilisateur (DP)
C       BSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X donnees par l'utilisateur (DP)
C       X : tableau de longueur egale au nombre de variables , contenant
C           l'optimum trouve a l'iteration presente (DP)
C
C       Parametres de sortie:
C       --------------------
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X corrigees par la routine (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X corrigees par la routine (DP)
C       MODIF : indique si une modification des bornes a ete effectuee.
C               Si c'est le cas , MODIF a la valeur .TRUE. et sinon , MODIF
C               a la valeur .FALSE. (LOG)
C
```

```
C       Specifications:
C       --------------
C
        INTEGER ITER,NV
        DOUBLE PRECISION EPS,XINF,XSUP,X,BINF,BSUP
        LOGICAL MODIF
        DIMENSION XINF(NV),XSUP(NV),X(2*NV),BINF(NV),BSUP(NV)
C
C       Declarations:
C       ------------
C
        INTEGER IND
C
C       Programme:
C       ---------
C
        MODIF=.FALSE.
        DO 100 IND=1,NV
          IF(ABS(X(IND+MOD(ITER,2)*NV)-XINF(IND)).LT.EPS)THEN
            IF(XINF(IND).NE.BINF(IND))THEN
              XINF(IND)=XINF(IND)-50.
              MODIF=.TRUE.
            ENDIF
          ELSEIF(ABS(X(IND+MOD(ITER,2)*NV)-XSUP(IND)).LT.EPS)THEN
            IF(XSUP(IND).NE.BSUP(IND))THEN
              XSUP(IND)=XSUP(IND)+50.
              MODIF=.TRUE.
            ENDIF
          ENDIF
100     CONTINUE
        RETURN
        END
C
```

```
C     ************************************************************
C
      SUBROUTINE CYCLE(NV,BINF,BSUP,XINF,XSUP,MODIF)
C     ==============================================
C
C     Role:en cas de cyclage du a un mauvais choix des bornes fictives ,
C     ---- celles-ci sont ajustees i.e. l'intervalle est retreci.
C          La variable MODIF est mise a .TRUE. si un tel ajustement
C          s'est avere necessaire , sinon elle a la valeur .FALSE.
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       NV : nombre de variables (INT)
C       BINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X donnees par l'utilisateur (DP)
C       BSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X donnees par l'utilisateur (DP)
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X courantes (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X courantes (DP)
C
C       Parametres de sortie:
C       --------------------
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X corrigees par la routine (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X corrigees par la routine (DP)
C       MODIF : indique si une modification des bornes a ete effectuee.
C               Si c'est le cas , MODIF a la valeur .TRUE. et sinon , MODIF
C               a la valeur .FALSE. (LOG)
C
```

```
C
C         Specifications:
C         --------------
C
          INTEGER NV
          DOUBLE PRECISION BINF,BSUP,XINF,XSUP
          LOGICAL MODIF
          DIMENSION BINF(NV),BSUP(NV),XINF(NV),XSUP(NV)
C
C         Declarations:
C         ------------
C
          INTEGER IND
C
C         Programme:
C         ---------
C
          MODIF=.FALSE.
          DO 100 IND=1,NV
            IF(BINF(IND).NE.XINF(IND))THEN
              XINF(IND)=XINF(IND)+ABS(XSUP(IND)-XINF(IND))/4.
              MODIF=.TRUE.
            ENDIF
            IF(BSUP(IND).NE.XSUP(IND))THEN
              XSUP(IND)=XSUP(IND)-ABS(XSUP(IND)-XINF(IND))/3.
              MODIF=.TRUE.
            ENDIF
100       CONTINUE
          RETURN
          END
C
```

```
C     ************************************************************************
C
      SUBROUTINE INITBO(NV,X,BINF,BSUP,XINF,XSUP)
C     ===========================================
C
C     Role : La routine initialise par defaut les bornes inexistantes.
C     ----
C
C     Documentation:
C     -------------
C       Parametres d'entree:
C       -------------------
C       NV : nombre de variables (INT)
C       X : tableau de longueur egale au nombre de variables contenant
C           le point de depart (INT)
C       BINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X donnees par l'utilisateur (DP)
C       BSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X donnees par l'utilisateur (DP)
C
C       Parametres de sortie:
C       --------------------
C       XINF : tableau de longueur egale au nombre de variables , contenant
C              les bornes inferieures sur X completees par la routine (DP)
C       XSUP : tableau de longueur egale au nombre de variables , contenant
C              les bornes superieures sur X completees par la routine (DP)
C
```

```
C
C       Specifications:
C       ---------------
C
        INTEGER NV
        DOUBLE PRECISION BINF,BSUP,XINF,XSUP,X
        DIMENSION BINF(NV),BSUP(NV),XINF(NV),XSUP(NV),X(NV)
C
C       Declarations:
C       -------------
C
        INTEGER IND
C
C       Programme:
C       ----------
C
        DO 100 IND=1,NV
          IF(BINF(IND).EQ.-10.E10)THEN
            XINF(IND)=-100.+X(IND)
            IF(XINF(IND).GE.BSUP(IND))THEN
              XINF(IND)=BSUP(IND)-100.
            ENDIF
          ELSE
            XINF(IND)=BINF(IND)
          ENDIF
          IF(BSUP(IND).EQ.10.E10)THEN
            XSUP(IND)=100.+X(IND)
            IF(XSUP(IND).LE.BINF(IND))THEN
              XSUP(IND)=BINF(IND)+100.
            ENDIF
          ELSE
            XSUP(IND)=BSUP(IND)
          ENDIF
100     CONTINUE
        RETURN
        END
C
```

```
      SUBROUTINE DERIV(FCN, X, N, INFO, VALEUR)
C     ========================================
C
C        ROLE: calcule une derivee d'une fonction en un point par difference
C        ----  finie.
C
C     .. Scalar Arguments ..
      INTEGER           INFO, N
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION VALEUR(N), X(N)
C     ..
C     .. Subroutine Arguments ..
      EXTERNAL          FCN
C     ..
C     .. Local Scalars ..
      DOUBLE PRECISION EPS, FMOINS, FPLUS
      INTEGER           I, J
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION XPRIM(100)
C     ..
C
      EPS       = 1.E-5
      DO 10 I = 1, N
        XPRIM(I) = X(I)
10    CONTINUE

      DO 20 J = 1, N
        XPRIM(J) = X(J) + EPS
        CALL FCN(XPRIM, FPLUS, N, INFO)
        XPRIM(J) = X(J) - EPS
        CALL FCN(XPRIM, FMOINS, N, INFO)
        VALEUR(J) = (FPLUS-FMOINS)/ (2.D+00*EPS)
        XPRIM(J) = X(J)
20    CONTINUE

      RETURN
      END
C
C
```

```
      SUBROUTINE EVAL1F(X, FVAL, N, KFORM)
C     ====================================
C
C        ROLE: evaluation d'une fonction en un point donne.
C        ----
C
C MAXLEN : FORMULA  LENGTH              (255)
C LENSYM : SYMBOL   LENGTH              (1+5)
C LENVAR : MAXIMUM  VARIABLE LENGTH     (1+8)
C LENERR : ERROR MESSAGE LENGTH         (50)
C     .. Parameters ..
      INTEGER           MAXLEN, LENSYM, LENVAR, LENERR, NBFOMX
      PARAMETER         (MAXLEN=255, LENSYM=6, LENVAR=9, LENERR=50,
     1                  NBFOMX=101)
C     ..
C     .. Local Scalars ..
      REAL              RCUR
      INTEGER           I, ICUR, IERROR
      LOGICAL           INTRES
      CHARACTER         ERREUR* (LENERR)
C     ..
C     .. Local Arrays ..
      REAL              VARVAL(MAXLEN)
      INTEGER           CODTMP(MAXLEN)
C     ..

C     .. Scalar Arguments ..
      DOUBLE PRECISION  FVAL
      INTEGER           KFORM, N
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION  X(N)
C     ..
C     .. Arrays in Common ..
      REAL              PARLIS(MAXLEN)
      INTEGER           CODE(MAXLEN, NBFOMX), PARTYP(MAXLEN),
     1                  VARTYP(MAXLEN)
C     ..
C     .. External Subroutines ..
      EXTERNAL          OMFEEV
C     ..
C     .. Common blocks ..
      COMMON            /CCODE/CODE, PARLIS, PARTYP, VARTYP
C     ..
C
C
```

```
C
      DO 10 I = 1, N
        VARVAL(I) = X(I)
10    CONTINUE

      DO 20 I = 1, MAXLEN
        CODTMP(I) = CODE(I, KFORM)
20    CONTINUE
30    CONTINUE


C ---      EVALUATION DES RESULTATS
      CALL OMFEEV(CODTMP, PARLIS, PARTYP, VARVAL, VARTYP, RCUR, ICUR,
     1            INTRES, IERROR, ERREUR)


      IF (INTRES) THEN
        FVAL    = ICUR
      ELSE
        FVAL    = RCUR
      END IF


      RETURN
      END
C
C
```

```
      SUBROUTINE GUESS(VARNAM, VALUE)
C     ===============================
C
C        ROLE: initialisation du point de depart par l'utilisateur.
C        ----
C
C     .. Parameters ..
      INTEGER          MAXLEN, LENVAR, MAXVAR, MAXCON
      PARAMETER        (MAXLEN=255, LENVAR=9, MAXVAR=100, MAXCON=100)
C     ..
C     .. Scalar Arguments ..
      REAL             VALUE
      CHARACTER        VARNAM* (*)
C     ..
C     .. Scalars in Common ..
      INTEGER          NBVLIS
C     ..
C     .. Arrays in Common ..
      DOUBLE PRECISION FCHAP(MAXCON), X(2*MAXVAR), Z(MAXCON)
      CHARACTER        VARLIS(MAXLEN)* (LENVAR)
C     ..
C     .. Local Scalars ..
      INTEGER          KVAR
C     ..
C     .. External Functions ..
      INTEGER          OASTRA
      EXTERNAL         OASTRA
C     ..
C     .. Common blocks ..
      COMMON           /CVAR/NBVLIS, VARLIS
      COMMON           /DON2/X, FCHAP, Z
C     ..
      KVAR     = OASTRA(VARNAM, VARLIS, NBVLIS, 7)
      IF (KVAR.GT.0) X(KVAR)  = VALUE

      RETURN
      END
C
C
```

```
C       ***************************************************************************
C
        SUBROUTINE PACK(LINE)
C       =====================
C
C
C       Specifications:
C       --------------
C
        CHARACTER*(*) LINE
C
C       Declarations:
C       -------------
C
        INTEGER N,I,IN
C
C       Programme:
C       ---------
C
        N=LEN(LINE)
        IN=0
        DO 10 I=1,N
          IF(LINE(I:I).NE.' ')THEN
            IN=IN+1
            LINE(IN:IN)=LINE(I:I)
          ENDIF
10      CONTINUE
        IF(IN.LT.N) LINE(IN+1:N)=' '
        RETURN
        END
```

```
      SUBROUTINE PRFCN(NV, NC)
C     ========================
C
C
C        ROLE: routine generale s'occupant de la lecture des donnees.
C        ----
C
C     .. Parameters ..
      INTEGER           MAXLEN, LENSYM, LENVAR, LENERR, NBFOMX
      PARAMETER         (MAXLEN=255, LENSYM=6, LENVAR=9, LENERR=50,
     1                  NBFOMX=101)
C     ..
C     .. Local Scalars ..
      REAL              TMP
      INTEGER           IE, INDCOM, IS, NBCOM
      CHARACTER         DELIM*1, LINE*80, WORD*80
C     ..
C     .. Local Arrays ..
      CHARACTER         VARNAM(MAXLEN)* (LENVAR), COMLIS(4)* (5)
C     ..
C
C     .. Scalar Arguments ..
      INTEGER           NC, NV
C     ..

C     .. External Functions ..

      INTEGER           OASTLU, OASTRA
      EXTERNAL          OASTRA, OASTLU
C     ..
C     .. External Subroutines ..
      EXTERNAL          GUESS, OASTGW, OASTJU, SLB, SUB
C     ..

C
C
```

```
C
      NBCOM     = 4
      COMLIS(1) = 'SLB'
      COMLIS(2) = 'SUB'
      COMLIS(3) = 'GUESS'
      COMLIS(4) = 'MODEL'

10    CONTINUE
C ---     LECTURE D'UNE NOUVELLE LIGNE
      READ (1, FMT='(A)', END=20) LINE
C        CALL PACK(LINE)
      CALL OASTJU(LINE, 1)

C ---     ANALYSE DE LA LIGNE
      IS        = 0
      IE        = OASTLU(LINE)
      CALL OASTGW(LINE, IS, IE, ' ', WORD, DELIM,IERR)
      INDCOM    = OASTRA(WORD, COMLIS, NBCOM, 7)
      IF (INDCOM.LE.0) GO TO 10
      CALL OASTGW(LINE, IS, IE, ' ', VARNAM, DELIM,IERR)
      READ (LINE(IS+1:IE), FMT='(F13.0)') TMP

C ---     DETERMINATION DU TYPE DE LIGNE

      IF (INDCOM.EQ.1) THEN
        CALL SLB(VARNAM, TMP)
      ELSE IF (INDCOM.EQ.2) THEN
        CALL SUB(VARNAM, TMP)
      ELSE IF (INDCOM.EQ.3) THEN
        CALL GUESS(VARNAM, TMP)
      ELSE IF (INDCOM.EQ.4) THEN
        CALL RDMODEL(NV,NC)
      END IF

      GO TO 10
C
20    CONTINUE

      RETURN
      END
C
C
```

```
      SUBROUTINE RDMODEL(NV, NC)
C     ==========================
C        ROLE: traite le cas d'une ligne du type 'MODEL' i.e. lit la fonction
C        ----  objectif et les contraintes , et les traduit en formules codees
C              en tenant a jour une liste des variables et des parametres.
C
C MAXLEN : FORMULA  LENGTH              (255)
C LENSYM : SYMBOL   LENGTH              (1+5)
C LENVAR : MAXIMUM  VARIABLE LENGTH     (1+8)
C LENERR : ERROR MESSAGE LENGTH         (50)
C     .. Parameters ..
      INTEGER           MAXLEN, LENSYM, LENVAR, LENERR, NBFOMX
      PARAMETER         (MAXLEN=255, LENSYM=6, LENVAR=9, LENERR=50,
     1                  NBFOMX=101)
C     ..
C     .. Local Scalars ..
      REAL              TMP
      INTEGER           I, IE, IERROR, IS, J, K, KFORM, NBFORM, NBPAR,
     1                  NBPLIS, NBVAR
      CHARACTER         DELIM*1, LINE*80, WORD*80, ERREUR* (LENERR)
C     ..
C     .. Local Arrays ..
      REAL              PARVAL(MAXLEN), VARVAL(MAXLEN)
      INTEGER           IPLIS(MAXLEN), IVLIS(MAXLEN), NBRCOD(NBFOMX),
     1                  PARTYPTMP(MAXLEN)
      CHARACTER         FORMUL(NBFOMX)* (MAXLEN),
     1                  PARNAM(MAXLEN)* (LENVAR),
     2                  VARNAM(MAXLEN)* (LENVAR)
C     ..
C
C     .. Scalar Arguments ..
      INTEGER           NC, NV
C     ..
C     .. Arrays in Common ..
      REAL              PARLIS(MAXLEN)
      INTEGER           CODE(MAXLEN, NBFOMX), PARTYP(MAXLEN),
     1                  VARTYP(MAXLEN)
      CHARACTER         VARLIS(MAXLEN)* (LENVAR)
C     ..
C     .. External Subroutines ..
      EXTERNAL          OASTGW, OASTJU, OMFMPF, SETFCHAP
C     ..
C     .. Common blocks ..
      COMMON            /CCODE/CODE, PARLIS, PARTYP, VARTYP
      COMMON            /CVAR/NBVLIS, VARLIS
C     ..
C     .. Scalars in Common ..
      INTEGER           NBVLIS
C     ..
C     .. External Functions ..
      INTEGER           OASTLU
      EXTERNAL          OASTLU
C     ..
C
```

```
C
      NBFORM   = 1
      FORMUL(1) = '1'

10    CONTINUE

C ---     LECTURE D'UNE NOUVELLE LIGNE
      READ (1, FMT='(A)', END=20) LINE
C        CALL PACK(LINE)
C ---     JUSTIFICATION A GAUCHE
      CALL OASTJU(LINE, 1)
      IF (LINE.EQ.'END      ') GO TO 20

C ---     ANALYSE DE LA LIGNE: DETERMINATION DU TYPE DE LIGNE
      IS       = 0
      IE       = OASTLU(LINE)
      CALL OASTGW(LINE, IS, IE, '=<>', WORD, DELIM,IERR)
      IF (DELIM.EQ.'=') THEN

C ---     IL S'AGIT DE LA FONCTION OBJECTIF

        IF (WORD.EQ.'MIN') THEN
          FORMUL(1) = LINE(IS+1:IE)
        ELSE IF (WORD.EQ.'MAX') THEN
          FORMUL(1) = '-1*('//LINE(IS+1:IE)//')'
        END IF

C ---     IL S'AGIT D'UNE CONTRAINTE

      ELSE IF (DELIM.EQ.'<') THEN
        NBFORM   = NBFORM + 1
        FORMUL(NBFORM) = WORD
        READ (LINE(IS+1:IE), FMT='(F13.0)') TMP
        CALL SETFCHAP(NBFORM-1, TMP)
      ELSE IF (DELIM.EQ.'>') THEN
        NBFORM   = NBFORM + 1
        FORMUL(NBFORM) = '-1*('//WORD//')'
        READ (LINE(IS+1:IE), FMT='(F13.0)') TMP
        CALL SETFCHAP(NBFORM-1, -TMP)
      END IF

      GO TO 10
C
20    CONTINUE

      NBVLIS   = 0
      NBPLIS   = 0

      DO 90 KFORM = 1, NBFORM

C ---     ANALYSE DE LA FORMULE
        NBVAR    = 0
        NBPAR    = 0
        CALL OMFMPF(FORMUL(KFORM), CODE(1,KFORM), NBRCOD(KFORM), VARNAM,
     1              PARNAM, VARTYP, PARTYPTMP, VARVAL, PARVAL, NBVAR,
     2              NBPAR, IERROR, ERREUR)

        DO 30 I = 1, NBVAR
          VARVAL(I) = I
30      CONTINUE

C
```

```
C ---   UPDATE DE LA LISTE DES VARIABLES ET CONSTANTES

C PARAMETRES
        DO 50 I = 1, NBPAR
          IPLIS(I) = NBPLIS + I
          PARLIS(IPLIS(I)) = PARVAL(I)
          PARTYP(IPLIS(I)) = PARTYPTMP(I)
50      CONTINUE
        NBPLIS   = NBPLIS + NBPAR

C VARIABLES
        DO 70 I = 1, NBVAR
          J         = 0
          DO 60 K = 1, NBVLIS
            IF (VARNAM(I).EQ.VARLIS(K)) J         = K
60        CONTINUE
          IF (J.GT.0) THEN
            IVLIS(I) = J
          ELSE
            NBVLIS   = NBVLIS + 1
            VARLIS(NBVLIS) = VARNAM(I)
            IVLIS(I) = NBVLIS
          END IF
70      CONTINUE

C ---  SUBSTITUER DANS LE CODE
        DO 80 I = 8 + 1, 8 + NBRCOD(KFORM)
          IF (CODE(I,KFORM).LE.1000) THEN
            CODE(I, KFORM) = IPLIS(CODE(I,KFORM))
          ELSE IF (CODE(I,KFORM).GT.1000 .AND. CODE(I,KFORM).LE.
     1              2000) THEN
            CODE(I, KFORM) = IVLIS(CODE(I,KFORM)-1000) + 1000
          END IF
80      CONTINUE

90    CONTINUE


      DO 100 I = 1, NBVLIS
        VARTYP(I) = 1
        VARVAL(I) = I
100   CONTINUE


      NV        = NBVLIS
      NC        = NBFORM - 1

      RETURN
      END
C
C
```

```
      SUBROUTINE SETFCHAP(K, VALUE)
C     ============================
C
C        ROLE: initialisation des seconds membres des contraintes.
C        ----
C
C     .. Parameters ..
      INTEGER              MAXLEN, LENVAR, MAXVAR, MAXCON
      PARAMETER            (MAXLEN=255, LENVAR=9, MAXVAR=100, MAXCON=100)
C     ..
C     .. Scalar Arguments ..
      REAL                 VALUE
      INTEGER              K
C     ..
C     .. Arrays in Common ..
      DOUBLE PRECISION FCHAP(MAXCON), X(2*MAXVAR), Z(MAXCON)
C     ..
C     .. Common blocks ..
      COMMON               /DON2/X, FCHAP, Z
C     ..
      FCHAP(K) = VALUE
      RETURN
      END

C
```

```
      SUBROUTINE SLB(VARNAM, VALUE)
C     ============================
C
C         ROLE: initialisation des bornes inferieures par l'utilisateur.
C         ----
C
C     .. Parameters ..
      INTEGER          MAXLEN, LENVAR, MAXVAR, MAXCON
      PARAMETER        (MAXLEN=255, LENVAR=9, MAXVAR=100, MAXCON=100)
C     ..
C     .. Scalar Arguments ..
      REAL             VALUE
      CHARACTER        VARNAM* (*)
C     ..
C     .. Scalars in Common ..
      INTEGER          NBVLIS
C     ..
C     .. Arrays in Common ..
      DOUBLE PRECISION BINF(MAXVAR), BSUP(MAXVAR), FCHAP(MAXCON),
     1                 X(2*MAXVAR), Z(MAXCON)
      CHARACTER        VARLIS(MAXLEN)* (LENVAR)
C     ..
C     .. Local Scalars ..
      INTEGER          KVAR
C     ..
C     .. External Functions ..
      INTEGER          OASTRA
      EXTERNAL         OASTRA
C     ..
C     .. Common blocks ..
      COMMON           /CVAR/NBVLIS, VARLIS
      COMMON           /CDON1/BINF, BSUP
C     ..
      KVAR     = OASTRA(VARNAM, VARLIS, NBVLIS, 7)
      IF (KVAR.GT.0) BINF(KVAR) = VALUE

      RETURN
      END
C
C
```

```
      SUBROUTINE SUB(VARNAM, VALUE)
C     ============================
C
C
C        ROLE: initialisation des bornes superieures par l'utilisateur.
C        ----
C
C     .. Parameters ..
      INTEGER          MAXLEN, LENVAR, MAXVAR, MAXCON
      PARAMETER        (MAXLEN=255, LENVAR=9, MAXVAR=100, MAXCON=100)
C     ..
C     .. Scalar Arguments ..
      REAL             VALUE
      CHARACTER        VARNAM* (*)
C     ..
C     .. Scalars in Common ..
      INTEGER          NBVLIS
C     ..
C     .. Arrays in Common ..
      DOUBLE PRECISION BINF(MAXVAR), BSUP(MAXVAR)
      CHARACTER        VARLIS(MAXLEN)* (LENVAR)
C     ..
C     .. Local Scalars ..
      INTEGER          KVAR
C     ..
C     .. External Functions ..
      INTEGER          OASTRA
      EXTERNAL         OASTRA
C     ..
C     .. Common blocks ..
      COMMON           /CVAR/NBVLIS, VARLIS
      COMMON           /CDON1/BINF, BSUP
C     ..
      KVAR     = OASTRA(VARNAM, VARLIS, NBVLIS, 7)
      IF (KVAR.GT.0) BSUP(KVAR) = VALUE
      RETURN
      END
C
C
```

Références :

[1] Schmit, L.A., and Farshi, B., "Some approximation concepts for structural synthesis", AIAA Journal, volume 12, 1974, pp. 692 - 699.

[2] Fleury, C., and Braibant, V., "Structural optimization. A new dual method using mixed variables", LTAS Report SA-115, University of Liège, 1984.

[3] Nguyen, V.H., Strodiot, J.J., and Fleury, C., "A mathematical convergence analysis of the convex linearization method for engineering design optimization", Engineering optimization, à paraître, 1987.

[4] Svanberg, K., "Method of moving asympots. A new method for structural optimization", International Journal for numerical methods in Engineering, à paraître, 1986.

[5] Zangwill, "Non linear programming : a unified approach", Prentice - Hall, Inc, Englewood Cliffs, N. J 1969

[6] Kazesaj, M., "Modélisation et résolution de problèmes d'optimisation non linéaire de grande taille", thèse 1985, Université des Sciences et Techniques de Lille.

[7] Huard, P., "Point - to - set maps and mathematical programming", Mathematical programming study, volume 10, 1979, North - Holland Publishing company.

[8] Huard, P., "Une approche synthétique des recherches linéaires dans les méthodes de pente", EDF, bulletin de la direction des études et recherches, n° 2, pp 82 - 132, 1982.

[9] Hock, W., and Schittkowski, K., "Test examples for non linear programming codes", Lecture notes in Economics and mathematical systems, 187, Springer-Verlag Berlin Heidelberg New-York, 1981.

[10] Haftka, R.T., and Kamat, M.P., "Elements of structural optimization", Martinus Nijhoff Publishers, 1985.